MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

A.I. TR No. 1527 January, 1995

Modeling flue pipes: subsonic flow, lattice Boltzmann, and parallel distributed computers

Panayotis A. Skordos

This publication can be retrieved by anonymous ftp to publications.ai.mit.edu.

Abstract

The problem of simulating the hydrodynamics and the acoustic waves inside wind musical instruments such as the recorder, the organ, and the flute is considered. The problem is attacked by developing suitable local-interaction algorithms and a parallel simulation system on a cluster of non-dedicated workstations. Physical measurements of the acoustic signal of various flue pipes show good agreement with the simulations. Previous attempts at this problem have been frustrated because the modeling of acoustic waves requires small integration time steps which make the simulation very compute-intensive. In addition, the simulation of subsonic viscous compressible flow at high Reynolds numbers is susceptible to slow-growing numerical instabilities which are triggered by high-frequency acoustic modes.

The numerical instabilities are mitigated by employing suitable explicit algorithms: lattice Boltzmann method, compressible finite differences, and fourth-order artificial-viscosity filter. Further, a technique for accurate initial and boundary conditions for the lattice Boltzmann method is developed, and the second-order accuracy of the lattice Boltzmann method is demonstrated.

The compute-intensive requirements are handled by developing a parallel simulation system on a cluster of non-dedicated workstations. The system achieves 80% parallel efficiency (speedup/processors) using 20 HP-Apollo workstations. The system is built on UNIX and TCP/IP communication routines, and includes automatic process migration from busy hosts to free hosts.

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-92-J-4097 and by the National Science Foundation under grant number 9001651-MIP.

Modeling flue pipes: subsonic flow, lattice Boltzmann, and parallel distributed computers

by

Panayotis A. Skordos

B.S., Massachusetts Institute of Technology (1986)
S.M., Massachusetts Institute of Technology (1988)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

January 1995

Signature of Author .....
Department of Electrical Engineering and Computer Science
January 31, 1995

Certified by .....
Gerald Jay Sussman
Matsushita Professor of Electrical Engineering
Thesis Supervisor

Accepted by .....
Frederic R. Morgenthaler
Chairman, Departmental Committee on Graduate Students

Modeling flue pipes: subsonic flow, lattice Boltzmann, and parallel distributed computers

by

Panayotis A. Skordos

Submitted to the Department of Electrical Engineering and Computer Science
on January 31, 1995, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

The problem of simulating the hydrodynamics and the acoustic waves inside wind musical instruments such as the recorder, the organ, and the flute is considered. The problem is attacked by developing suitable local-interaction algorithms and a parallel simulation system on a cluster of non-dedicated workstations. Physical measurements of the acoustic signal of various flue pipes show good agreement with the simulations. Previous attempts at this problem have been frustrated because the modeling of acoustic waves requires small integration time steps which make the simulation very compute-intensive. In addition, the simulation of subsonic viscous compressible flow at high Reynolds numbers is susceptible to slow-growing numerical instabilities which are triggered by high-frequency acoustic modes.

The numerical instabilities are mitigated by employing suitable explicit algorithms: lattice Boltzmann method, compressible finite differences, and fourth-order artificial-viscosity filter. Further, a technique for accurate initial and boundary conditions for the lattice Boltzmann method is developed, and the second-order accuracy of the lattice Boltzmann method is demonstrated.

The compute-intensive requirements are handled by developing a parallel simulation system on a cluster of non-dedicated workstations. The system achieves 80% parallel efficiency (speedup/processors) using 20 HP-Apollo workstations. The system is built on UNIX and TCP/IP communication routines, and includes automatic process migration from busy hosts to free hosts.

Thesis Supervisor: Gerald Jay Sussman
Title: Matsushita Professor of Electrical Engineering

Acknowledgments

Gerry Sussman and Hal Abelson have helped me throughout my studies at MIT with their support, advice, and criticism. Especially, Gerry deserves extra thanks for more reasons than I can list here. Furthermore, it was Sussman's idea that my fluid dynamics algorithms could be used to simulate flue pipes, and the physical measurements of the acoustic signal of flue pipes were performed with Gerry's help. Gary Doolen of the Los Alamos National Laboratory has generously provided support, advice, and hospitality during my summer visits at the lab where a significant part of this work was done.

This thesis is dedicated to my parents

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-92-J-4097 and by the National Science Foundation under grant number 9001651-MIP.

Biographical Note

Panayotis Skordos was born in Athens, Greece, on September 6, 1965. He came to the United States to attend high school for one year, and then entered the Massachusetts Institute of Technology in 1983. He received the S.B., the M.S., and the Ph.D. degrees from MIT in Electrical Engineering and Computer Science in 1986, 1988, and 1995 respectively. During this period, he also spent summers at the Institute of Advanced Study in Princeton, the Santa Fe Institute, and the Los Alamos National Laboratory.

His professional interests include parallel scientific computing, distributed systems, numerical methods, artificial intelligence in computing, and applications such as the simulation of fluid dynamics inside wind musical instruments. In the past, he has also worked on numerical methods for ordinary differential equations to model the orbits of the planets, and on computer simulations of microscopic systems of gas molecules to study the foundations of the second law of thermodynamics. In his spare time, he plays the piano, and keeps active in sports.

Contents

Chapter 1

Introduction

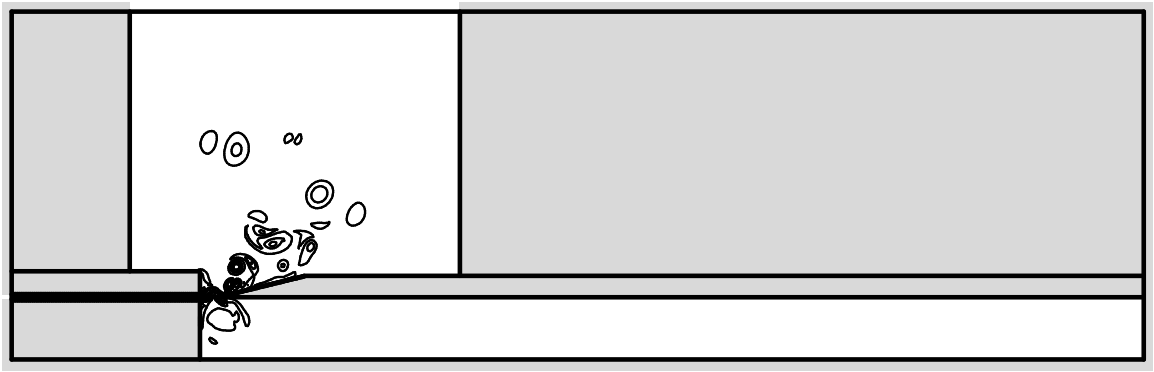

Figure 1-1: Simulation of a flue pipe that is 20 cm long, 1.34 cm wide, and produces tones near 400 and 1100 cycles per second. Air is blown through the flue at 1200 cm/s. Iso-vorticity contours are shown at 25 milliseconds after startup.

1.1 Thesis outline

I have considered the problem of simulating the hydrodynamics and the acoustic waves inside wind musical instruments such as the organ flue pipe. I have attacked this problem by developing suitable local-interaction algorithms and a parallel simulation system on a cluster of non-dedicated workstations. Previous attempts at this problem have been frustrated for two reasons: First, the modeling of acoustic waves requires small integration time steps which make the simulation very compute-intensive. Second, the simulation of subsonic viscous compressible flow at high Reynolds numbers

is susceptible to slow-growing numerical instabilities which are triggered by high-frequency acoustic modes.

Below, I outline the main results of my thesis, and I explain how my work fits in with previous work in computational fluid dynamics and in parallel computing. My contributions belong to three categories as follows:

- *Physical applications:* I demonstrate the first simulations of flue pipes ever-to-be-performed which model both hydrodynamics and acoustic waves together. Physical measurements of the acoustic signal of various flue pipes show good agreement with the simulations.

- *Numerical methods:* I mitigate the problem of numerical instabilities by employing a fourth-order artificial-viscosity filter. This filter can be used both with the lattice Boltzmann method and also with a compressible finite difference method. Further, I develop a technique for accurate boundary conditions and initial conditions for the lattice Boltzmann method, and I demonstrate the second-order accuracy of the lattice Boltzmann method.

- *Parallel computing:* I handle the problem of compute-intensive requirements by developing a parallel simulation system on a cluster of non-dedicated workstations. The system is based on local-interaction methods, small communication capacity, and automatic migration of parallel processes from busy hosts to free hosts. Typical simulations achieve 80% parallel efficiency (speedup/processors) using 20 HP-Apollo workstations.

Later in this chapter, I present a few representative simulations and physical measurements of the sound generated by a soprano recorder flue pipe. More simulations and measurements can be found in chapter 7. Between here and chapter 7, the technical crux of my thesis is presented. Specifically, the equations of fluid mechanics and fluid acoustics are reviewed in chapter 2. Numerical methods for simulating fluid

flow are analyzed in chapters 3 – 5. Parallel computing on a cluster of non-dedicated workstations is discussed in chapter 6.

Regarding numerical methods, I emphasize the lattice Boltzmann method because it is a new approach for simulating fluids which is promising, and is still undergoing refinements and improvements. I develop a technique for accurate initial and boundary conditions for the lattice Boltzmann method which is very important in practical situations. [1] Further, I demonstrate experimentally that the discretization error of the lattice Boltzmann method decreases quadratically with finer resolution both in space and in time. My results on the lattice Boltzmann method have been published in Skordos [48], and have helped to bring the lattice Boltzmann method from the physicists' world to the engineer's world.

Apart from the lattice Boltzmann method, I examine two different kinds of explicit finite difference methods. In chapter 4, I compare the lattice Boltzmann method against an incompressible finite difference method which neglects the acoustic waves and simulates incompressible flow. In chapters 6 and 7, I compare the lattice Boltzmann method against a compressible finite difference method which solves the compressible Navier Stokes equations. The lattice Boltzmann method appears to model acoustic waves slightly more accurately than the compressible finite difference method. However, my comparisons are not complete, and further work is needed to understand better the differences between the two approaches.

In general, I can say that the lattice Boltzmann approach has better stability properties than explicit finite difference methods because the lattice Boltzmann approach is based on relaxation as opposed to differencing operations. The ability of the lattice Boltzmann method to model acoustic waves well, which I mentioned above, is probably related to the stability properties and the smooth behavior of the lattice Boltzmann method for disturbances of small wavelength. A limitation of the lattice

[1] My technique also makes possible multigrids and interpolation between different grids for the lattice Boltzmann method (see section 4.6.2); however, I have not tested multigrids in actual simulations yet.

Boltzmann approach is that it can not handle arbitrary non-uniform grids. This limitation may be overcome to some extent by joining grids of different resolution (see my technique for boundary conditions), but this is a subject for future research. Here, I employ uniform grids only because they are simple to program, to understand, and to use in parallel computation.

1.2 Unexplored area of fluid dynamics

The simulation of fluid flow is very important for engineering and science because fluid phenomena can be found everywhere, in the sky, in the sea, inside engines, inside our bodies. Thus, there is great motivation for simulating fluids. On the other hand, the simulation of fluid phenomena is difficult because the equations of motion (known as the Navier Stokes equations) are nonlinear partial differential equations that exhibit a wide range of dynamical behavior and have no exact solutions in most cases. In addition, the simulation of fluid phenomena requires large amounts of data to represent the geometry and the dynamics of the flow accurately. Consequently, computers are challenged to their limits when simulating fluid flow, and there is a never-ending demand for increased computing power to enable finer and more realistic simulations.

So far, the field of computational fluid dynamics has succeeded in simulating flows of many different types: supersonic, transonic, flow through porous media, mixtures of fluids, free surface flows. In addition, progress has been made towards faithful simulation of turbulent flows and flows with chemical reactions. Yet, these achievements are only the beginning of a long exploration. As computer technology improves and new algorithms are discovered, more fluid phenomena will succumb to simulation. For instance, fluid phenomena that include two different time-scales, slow-moving hydrodynamics and fast-moving acoustic waves, are now possible to simulate numerically using parallel computers, as I demonstrate in my thesis. This is an area

of computational fluid dynamics that has remained unexplored until now.

The generation of sound inside wind musical instruments such as the organ, the recorder, and the flute is a phenomenon which depends on the interaction between hydrodynamics and acoustic waves. Specifically, when a jet of air impinges a solid obstacle in the vicinity of a cavity, the jet begins to oscillate strongly and produces acoustic waves. The acoustic waves reflect off the cavity, and return to interact with the jet according to a complex nonlinear feedback cycle. Similar phenomena that depend on the interaction between acoustic waves and jets occur in human whistling and in voicing of fricative consonants (Shadle85 [46]). The computer simulation of these phenomena provides a precise way of studying the phenomena and experimenting with different parameters.

The main difficulties that have prevented simulations of subsonic flow inside flue pipes arise from the fact that the subsonic flow involves two different time-scales, hydrodynamics and acoustic waves, which interact with each other nonlinearly. On the one hand, the simulation is compute-intensive because the integration time step must be very small to follow the acoustic waves (section 3.2.1). On the other hand, the simulation of compressible flow is susceptible to slow-growing numerical instabilities when the Reynolds number is large. I handle the compute-intensive requirements by developing a parallel simulation system on a cluster of workstations. In addition, I mitigate the numerical instabilities by employing a fourth-order artificial-viscosity filter (chapter 5) in combination with the lattice Boltzmann method and also in combination with a compressible finite difference method.

The traditional approach of simulating subsonic flow is to approximate the subsonic flow with a perfectly incompressible flow, as defined in section 2.4.3. The incompressible flow approximation ignores the propagation of acoustic waves (it assumes infinitely fast propagation), and allows the use of large integration time steps (Peyret&Taylor [38]). Such an approach is valid when the acoustic waves play a secondary role from a physical point of view: for example, when the time-scale of

acoustic waves does not influence the main flow, and when we are not interested in the generation of acoustic waves. The incompressible flow approach is also valid when we are interested in the generation of acoustic waves, but the acoustic waves do not interact with the hydrodynamics. In such a case, the incompressible flow solution can be computed separately and then used as a source term to the wave equation (Harding [24]). Moreover, the wave equation can be linearized, and can be solved using analytic approximations (Green function integrals, for example) avoiding the cost of a direct numerical solution.

The incompressible flow approximation is a good idea when the propagation of acoustic waves does not influence the dynamics of the phenomenon. However, it is inappropriate when the flow problem depends on the interaction between hydrodynamics and acoustic waves (the flow of air inside flue pipes, for example). The only way to simulate correctly such a flow is to simulate both the hydrodynamics and the acoustic waves together. In other words, the only way to simulate such a problem is to solve numerically the compressible Navier Stokes equations, and to compute the time-dependent evolution of the flow and the acoustic waves. This is the subject of my thesis.

1.3 Local-interaction parallel computing

Parallel computing is necessary in order to perform high resolution simulations of hydrodynamics and acoustic waves. To this end, I have developed a parallel system on a cluster of 25 non-dedicated workstations. The system achieves concurrency by decomposing the simulated area into subregions and by assigning the subregions to parallel subprocesses on different workstations. The use of explicit numerical methods leads to small communication requirements. The parallel subprocesses automatically migrate from busy hosts to free hosts in order to exploit the unused cycles of non-dedicated workstations, and to avoid disturbing the regular users. The system achieves 80%

parallel efficiency (speedup/processors) using 20 HP-Apollo workstations in a cluster where there are 25 non-dedicated workstations total. [2]

In chapter 6, I describe the implementation of the parallel simulation system, and I present detailed measurements of the parallel efficiency (speedup/processors) of 2D and 3D simulations of fluid dynamics. Further, I develop a theoretical model of efficiency which fits closely the measurements. The measurements show that the shared-bus Ethernet network is adequate for two-dimensional simulations of fluid dynamics, but limited for three-dimensional ones. I expect that new technologies in the near future such as Ethernet switches, FDDI and ATM networks will make practical three-dimensional simulations of fluid dynamics on a cluster of workstations.

It is worth emphasizing that the success of my parallel simulation system depends considerably on the use of explicit methods. This is because explicit methods are completely parallelizable, and lead to small communication requirements which can be satisfied on a cluster of workstations. The disadvantage of explicit methods is that small integration time steps are required for numerical stability. However, the simulation of subsonic flow requires small integration time steps, anyways, to model the fast-moving acoustic waves. Thus, there is a match between the requirements of the problem and the requirements of explicit methods. In addition, there is a match between the problem, the algorithms, and the computer system.

In general, explicit methods are desirable for parallel computing when increasing

[2] A major motivation for developing parallel computing on a cluster of workstations has been the high availability of workstations compared to other parallel computers. At the Artificial Intelligence Laboratory and the Laboratory for Computer Science at MIT where I have done most of this work, there is a Connection Machine CM-5 with 128 processors, but the machine is time-shared by too many people. There are typically 10 users sharing the 128 processors on the average, which reduces the computation power to 12 processors per user at best. This processing power is not enough for my purposes.

The computational speed of an HP9000/715 workstation is approximately 3-4 times the computational speed of one processor of the CM-5. Thus, a distributed simulation using 20 HP9000/715 workstations is equivalent approximately to 60-80 processors of the CM-5 running in dedicated mode. Of course, this comparison only applies to special problems that have a small ratio of communication to computation. Other problems that have large communication requirements would not run efficiently on my distributed system. Such problems might run efficiently on a parallel computer such as the CM-5 that has a powerful communication network.

numbers of local processing units are available with minimum communication capacity between the processing units. Such computers may be widespread in the future; for instance, a future parallel computer may consist of millions of local processing units, each unit having the power of one of today's workstations. Communication is going to dominate the cost of such computers, and methods that minimize communication are going to be desirable. With this perspective in mind, the work presented herein for a cluster of 25 workstations, may have applications to future parallel computers as well.

1.3.1 Comparison with other work in parallel computing

The suitability of local-interaction algorithms for parallel computing on a cluster of workstations has been demonstrated in previous works, such as [7], [9], and elsewhere. Cap&Strumpen [7] present the PARFORM system and simulate the unsteady heat equation using explicit finite differences. Chase&et al. [9] present the AMBER system, and solve Laplace's equation using Successive Over-Relaxation. The present work emphasizes, and clarifies further the importance of local-interaction methods for parallel systems with small communication capacity. Furthermore, a real problem of science and engineering is solved using the present approach. The problem is the simulation of subsonic flow with acoustic waves inside wind musical instruments.

In the fluid dynamics community, little attention has been given so far to simulations of hydrodynamics and acoustic waves. The reason is that such simulations are very compute-intensive, and can be performed only when parallel systems such as the one described herein are available. Furthermore, the fluid dynamics community has generally shunned the use of explicit methods because explicit methods require small integration time steps (see section 3.2). With the increasing availability of parallel systems, explicit methods are now attracting more attention in all areas of computational fluid dynamics. The present work clearly reveals the power of explicit methods in one particular area, and should motivate further work in explicit methods and

local-interaction algorithms.

Regarding parallel efficiency (speedup/processors), the efficiency of my parallel simulation system is very good, 80% typically. My measurements of the efficiency (section 6.7) are more detailed than any other reference that I know, especially for the case of a shared-bus Ethernet network. I also develop a model of parallel efficiency in section 6.8, which is based on simple ideas that have been discussed previously, for example in Fox et al. [19] and elsewhere. I compare the predictions of this model against real measurements of the parallel efficiency.

Regarding the problem of using non-dedicated workstations, I handle this problem by employing automatic process migration from busy hosts to free hosts. An alternative approach which has been used elsewhere is the dynamic allocation of processor workload. In the present context, dynamic allocation means to enlarge and to shrink the subregions which are assigned to each workstation depending on the CPU load of the workstation (Cap&Strumpen [7]). Although this approach is important in various applications (Blumofe&Park [5]), it seems unnecessary for simulating fluid flow problems with static geometry. For such problems, it may be simpler and more effective to use fixed size subregions per processor, and to apply automatic migration of processes from busy hosts to free hosts. This approach has worked very well in the parallel simulations presented here.

Regarding the design of the parallel simulation system, I have aimed for simplicity. In particular, the special constraints of local-interaction problems and static decomposition have guided the design of the parallel system. The automatic migration of processes has been implemented in a straightforward manner because the system is very simple. The availability of a homogeneous cluster of workstations, and a common file system have also simplified the implementation, which is based on UNIX and TCP/IP communication routines. The approach presented here works well for spatially-organized computations which employ a static decomposition and local-interaction algorithms.

My thesis does not examine issues such as high-level parallel programming, parallel languages, and inhomogeneous clusters of workstations. Efforts along these directions are the PVM system (Sunderam [50]), the Linda system (Carriero [8]), the packages of (Kohn&Baden [30]) and (Chesshire&Naik [11]) that facilitate parallel decomposition, the Orca language for distributed computing (Bal&et al. [1]), etc.

1.4 Some simulation results

This section describes a few representative simulations and physical measurements of the musical tones generated by a soprano recorder flue pipe.

1.4.1 Flue pipe of a soprano recorder

The recorder is a ZEN-ON SB-DX soprano recorder, made in Japan, and commonly available in music stores. The recorder consists of three parts which are made out of plastic, and which connect together to make the recorder (see figure 1-2).

- The head of the recorder consists of the flue (narrow passage where the jet of air is formed), the labium (sharp edge which the jet impinges), and a short cylindrical pipe of length 6.1 cm and diameter 1.34 cm.
- The main pipe of the recorder is designed to attach to the head of the recorder. The main pipe is cylindrical, it tapers along its length, and includes finger-holes for playing different tones.
- The end-piece of the recorder is designed to attach to the end of the main pipe. The end-piece has a flaring shape, and includes one double-finger-hole for playing the lowest notes C and $C^{\#}$ of the recorder.

For the purpose of testing the basic phenomenon of tone generation by the recorder, the finger-holes and the tapering shape of the recorder are not necessary, and they

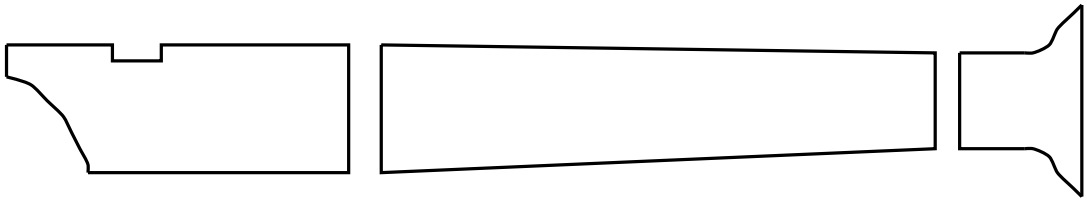

Figure 1-2: A three-piece soprano recorder.

are omitted here. Specifically, the main pipe of the recorder is replaced with a new pipe which has constant diameter and no finger-holes. The new pipe is connected to the head of the recorder which is 6.1 cm long. The addition of the new pipe results in lengths such as 20 cm which are typical of soprano recorders. It should be noted that the attached pipe has a slightly smaller diameter 1.27 cm than the head of the recorder 1.34 cm. This difference is very small, however, and is neglected in the computer simulations. The attached pipe is closed at the far end in the present experiments (see chapter 7 for simulations of open-end pipes).

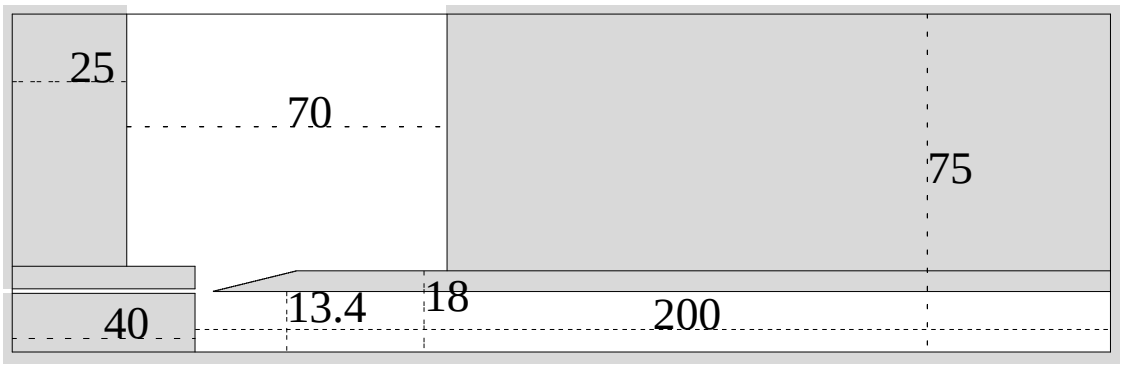

Figure 1-3: Soprano recorder flue, 20 cm pipe. The numbers shown correspond to millimeters.

Figures 1-3 and 1-4 show the recorder according to a 2D simplified geometry which is used in the simulations. The gray areas correspond to the walls around the recorder. The walls above the recorder are skipped in the simulation in order to reduce the computational effort. The pipe is located at the bottom of the picture, and measures

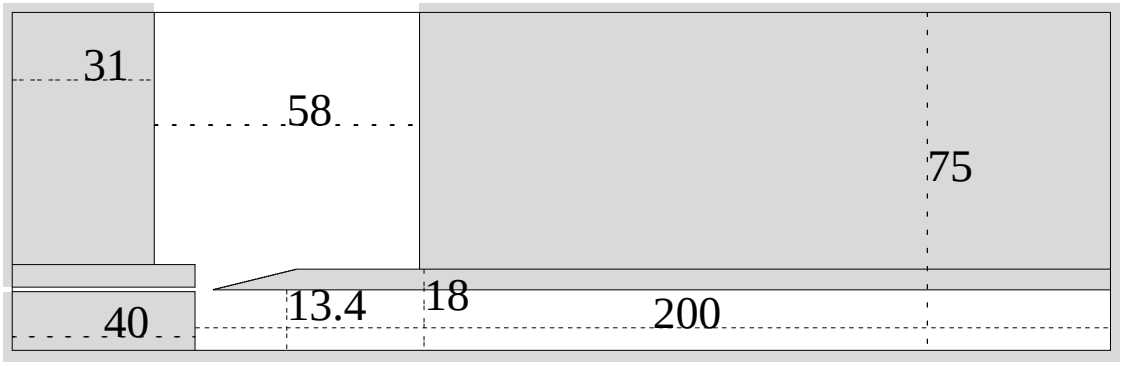

Figure 1-4: A smaller outlet region than figure 1-3.

20 cm long and 1.34 cm wide. The flue (or flue channel) is located at the bottom left corner, and measures 4 cm long and 0.1 cm wide. At a distance of 0.4 cm in front of the orifice of the flue (where the jet of air emerges), there is a sharp edge which is called the labium. The labium measures an angle of 14 degrees approximately, and is positioned slightly below the midline of the flue channel. Specifically, the tip of the labium is located at 1.34 cm from the bottom of the pipe, and the flue channel is located between 1.3 cm and 1.4 cm.

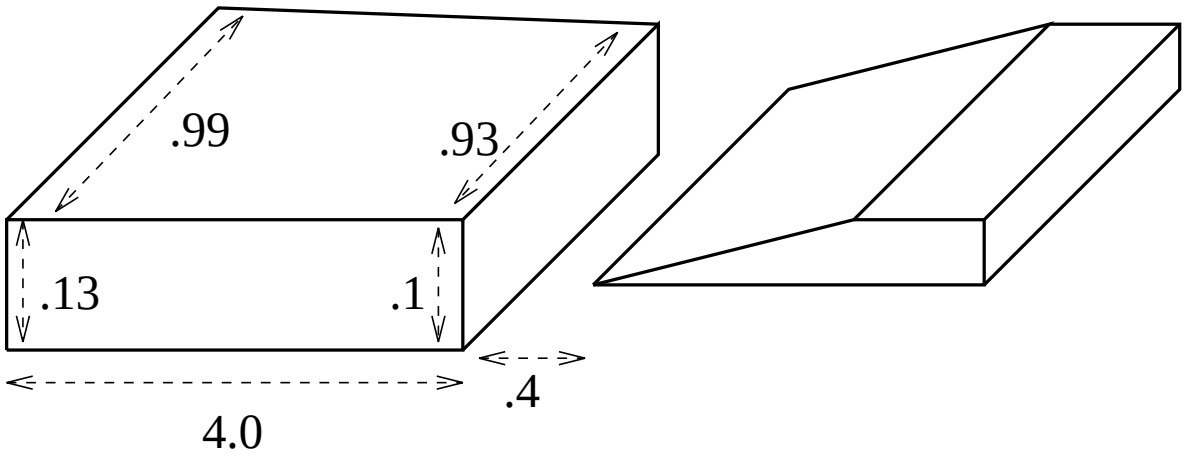

Figure 1-5: The flue and the labium in three dimensions. Not drawn to scale. The numbers correspond to centimeters.

In three-dimensions, the pipe of the recorder is a cylinder, and the flue channel and the labium are approximately rectangular as shown in figure 1-5. The flue channel is slightly curved along the sides which measure 0.99 cm and 0.93 cm, but the curvature is very small and is neglected here. Further, the flue channel tapers slightly along the side which measures 4.0 cm. Specifically, the flue channel measures 0.13 cm by 0.99 cm at the inlet (where air is blown into the recorder), and it measures 0.10 cm by 0.93 cm at the orifice (where the air emerges to strike the labium). The tapering of the flue channel is neglected in the computer simulations because it is very small.

The Reynolds number of the flow of air inside the soprano recorder ranges between 500 and 1700. The Reynolds number is defined as the mean speed of the jet of air inside the flue channel (typical speeds are between 800 and 2500 cm/s) times the width of the flue channel 0.1 cm, and divided by the kinematic viscosity of air 0.15 cm^2/s (see section 2.6 for details). High Reynolds numbers typically produce turbulent flow which involves very small length scales, and is difficult to simulate numerically. In the case of a narrow jet of air 0.1 cm wide, a Reynolds number above 500 is rather high, so that the jet is very unstable and becomes turbulent after exiting the orifice and impinging the labium. Although the computer simulations can not model the fine scales of turbulence (the grid size is only $\Delta x = 0.01$ cm), an artificial-viscosity filter is used which dissipates small-wavelengths in a pseudo-turbulent-like fashion (see section 5.5). It appears that a precise model of turbulence is not necessary to reproduce the basic operation of the flue pipe. Further investigation of this issue should be done in the future.

1.4.2 Computer simulations

Simulation results using the lattice Boltzmann method and the compressible finite difference method of section 3.3 are presented here. The simulations are based on the geometry shown in figure 1-3 for the lattice Boltzmann method, and on the geometry shown in figure 1-4 for the finite difference method. The two geometries are almost

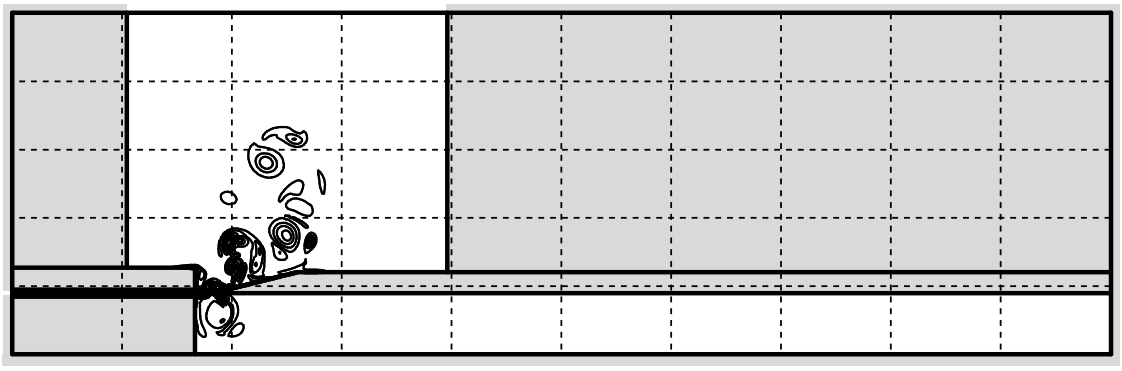

Figure 1-6: Simulation of a 20 cm flue pipe. The decomposition 10×6 is shown as dashed lines. 22 workstations are used. The gray-shaded areas are not simulated.

identical except that the outlet region is 8.0 cm wide in the former, and is 5.8 cm wide in the latter. The reason for this difference is purely accidental (availability of workstations), and is too small to affect the results significantly. However, it should be noted that very small outlet regions become quickly saturated with the vorticity generated by the flue, and complicate the simulation. Thus, the size of the outlet region should be as large as possible within one's computational constraints.

In the simulations, the air is forced through the inlet (the entrance of the flue channel), and exits through the outlet (the top part of the picture). During the initial blowing of the air into the flue channel, the imposed density and velocity at the inlet rise smoothly to final values within 3 ms (see section 7.3.2 for more details). Appropriate boundary conditions at the inlet and the outlet (see section 7.3) maintain the air flow through the recorder, and prevent reflection of acoustic waves at the inlet and the outlet. All other boundaries are solid walls and reflect the acoustic waves which are generated by the flue.

The spatial resolution of all the simulations presented in this section is $\Delta x = 0.01$ cm. This resolution corresponds to 10 fluid nodes along the width 0.1 cm of the flue channel (see figures 1-7 and 1-8), and produces adequate results. Finer-resolution simulations of flue pipes have also been performed (for example, 13 nodes along the

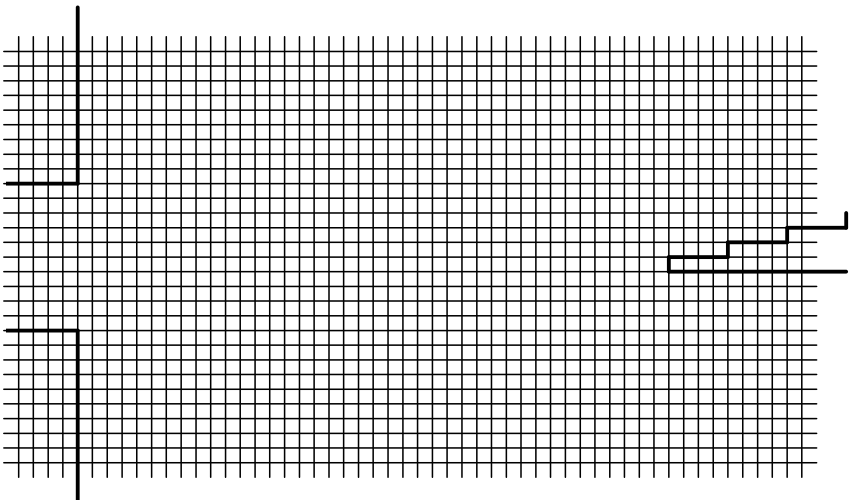

Figure 1-7: The grid at the flue-labium region, there are 10 fluid nodes along the width 0.1 cm of the flue channel.

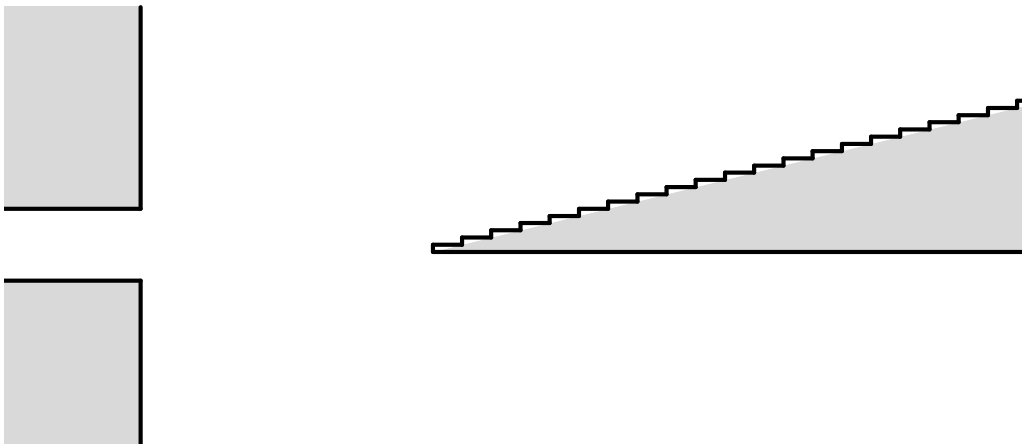

Figure 1-8: Magnified view of the orifice and the labium according to the simulations.

width of the flue channel), and the results do not change very much. Fewer than 10 nodes along the width of the flue channel are not recommended because the ratio of the width of the flue channel divided by the width of the tip of the labium (one Δx wide) should be at least $10:1$ in order to produce a "sharp" labium and in order to position the labium along the width of the flue channel with an accuracy of 0.01 cm.

The integration time step is determined from the requirement that the numerical speed $\Delta x/\Delta t$ must be of the order of the speed of sound $c_s = 34400$ cm/s. Accordingly, the time step is kept very small, for example $\Delta t = 2.1 \times 10^{-7}$ s, which makes the

Vmean cm/s	f_0 Hz	(λ_0) (cm)	A_0 10^{-5}	f_1 Hz	(λ_1) (cm)	A_1 10^{-6}	f_2 Hz	(λ_2) (cm)	A_2 10^{-6}
818	219	(157)	0.14	374	(92)	1.14	1159	(30)	0.71
1104	1132	(30)	1.88	395	(87)	5.07	1062	(32)	3.69
1535	1104	(31)	1.05	1873	(18)	8.82	387	(89)	7.49
1995	1926	(18)	3.56	417	(82)	18.7	1169	(29)	10.2

Table 1.1: Frequencies, lattice Boltzmann, 20 cm closed-end recorder

Vmean cm/s	f_0 Hz	(λ_0) (cm)	A_0 10^{-5}	f_1 Hz	(λ_1) (cm)	A_1 10^{-6}	f_2 Hz	(λ_2) (cm)	A_2 10^{-6}
838	424	(81)	0.12	326	(106)	1.01	1134	(30)	0.52
1113	1116	(31)	1.39	420	(82)	3.69	244	(141)	1.98
1634	1882	(18)	1.89	1182	(29)	7.85	329	(104)	6.58
2082	1957	(18)	4.26	377	(91)	25.1	1143	(30)	10.1

Table 1.2: Frequencies, compressible finite difference, 20 cm closed-end recorder

Vmean cm/s	f_0 Hz	(λ_0) (cm)	A_0 10^{-1}	f_1 Hz	(λ_1) (cm)	A_1 10^{-2}	f_2 Hz	(λ_2) (cm)	A_2 10^{-3}
734	395	(87)	1.051	1186	(29)	3.177	2768	(12)	10.55
1140	1111	(31)	1.095	401	(86)	8.754	1915	(18)	14.63
1558	1140	(30)	2.016	1879	(18)	0.996	398	(87)	7.557
1985	1145	(30)	2.676	3438	(10)	0.959	5730	(6)	6.169
2420	1918	(18)	2.947	3836	(9)	3.015	7670	(4.5)	0.889

Table 1.3: Frequencies, physical measurements, 20 cm closed-end recorder

simulation very compute-intensive, and makes parallel computing a necessity. Typical simulations correspond to 30 ms, and require 150000 integration steps. Figure 1-6 shows a typical decomposition of the geometry of a flue pipe into subregions for the purpose of parallel computing. The decomposition 10×5 is shown as dashed lines. The gray-shaded areas are not simulated, only the white areas are simulated. There are 22 rectangular subregions which are active, and are assigned to 22 workstations. Each workstation can update 39100 fluid nodes per second (when the lattice Boltzmann method is used, see chapter 6), and the parallel efficiency is approximately

20 cm pipe	f_0 Hz	(λ_0) (cm)	f_1 Hz	(λ_1) (cm)	f_2 Hz	(λ_2) (cm)	f_3 Hz	(λ_3) (cm)	f_4 Hz	(λ_4) (cm)
open-closed	430	(80)	1290	(26.7)	2150	(16)	3010	(11.4)	3870	(8.9)
open-open	860	(40)	1720	(20)	2580	(13.3)	3440	(10)	4300	(8)

Table 1.4: Ideal resonant frequencies, 20 cm, open-closed and open-open.

80%. It takes about 48 hours of running-time to perform 150000 integration steps using 0.79 million fluid nodes.

Figures 1-15 to 1-18 show acoustic signals obtained from simulations of the 20 cm closed-end recorder using the lattice Boltzmann method. Corresponding results using the compressible finite difference method are shown in figures 1-19 to 1-22. The major frequencies of the acoustic signals are summarized in tables 1.1 and 1.2. For comparison purposes, frequencies obtained from physical measurements are shown in table 1.3 (they are discussed in the next section), and the ideal resonant frequencies of a passive pipe 20 cm long are shown in table 1.4 (again explained in the next section).

A sampling interval of approximately 3.09×10^{-5} s is used in the computer simulations, which corresponds to a maximum frequency of 16.2 kHz. Frequencies of interest are less than 5 kHz, and are shown in the figures; frequencies higher than 5 kHz are not shown because they are of very small amplitude. Each figure plots the acoustic signal in the time domain at the bottom, and in the frequency domain at the top. In the time domain, the acoustic signal is shown as the relative density (a non-dimensional number). In the frequency domain, the acoustic signal is shown as the pressure normalized by a standard pressure level of 2×10^{-4} gm cm/s^2 (see section 2.6). Also, in the frequency domain the acoustic signal is plotted according to a logarithmic scale of $20 \log_{10}$ decibel (dB), so that a gain of 20 dB corresponds to a ratio of 10 in amplitude.

We notice that the computer simulations predict acoustic signals with amplitudes near 100 dB, which may seem too large for a recorder, but it should be noted that

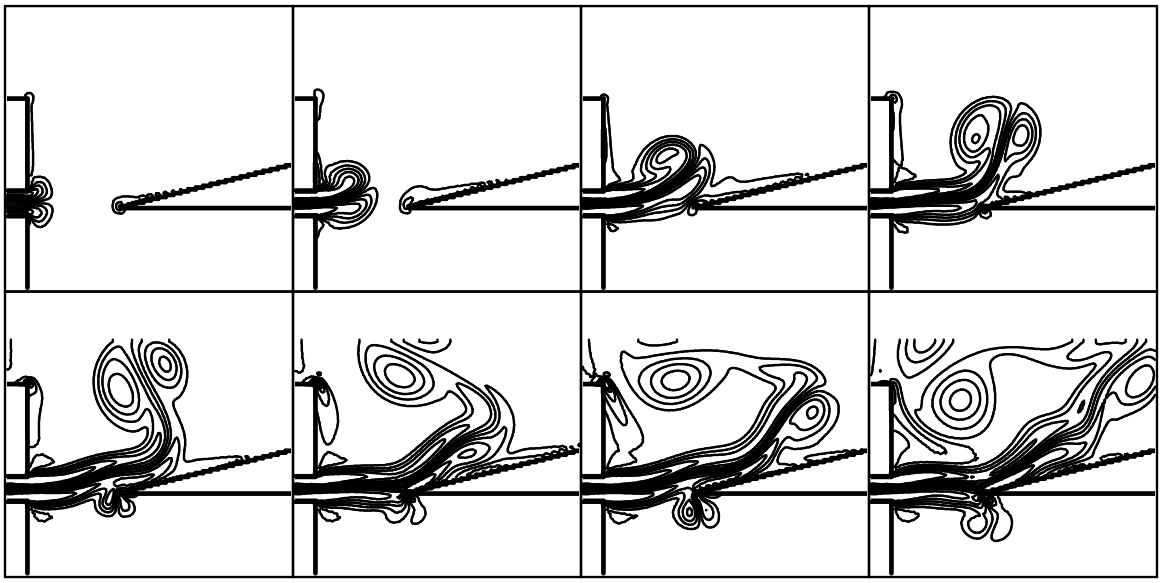

Figure 1-9: The flow during the initial blowing of air into the flue pipe. Frames are 0.49 ms apart, from left to right. Iso-vorticity contours are plotted.

the simulation is two-dimensional (the sound spreads as $1/r$ in 2D versus $1/r^2$ in 3D), and the acoustic signal is sampled inside a small outlet cavity very near the labium (approximately 5 cm above the labium). Thus, acoustic signals with amplitude near 100 dB are not surprising.

We also notice that the acoustic signals predicted by the lattice Boltzmann and the compressible finite difference methods are similar, but slightly different. Possible reasons for the differences are the following: The modeling of boundary conditions is different between lattice Boltzmann and finite differences because the computational structure of the methods is very different. Also, the lattice Boltzmann method can model the high-frequency components of acoustic waves more accurately than the compressible finite difference method. The above differences between the lattice Boltzmann method and the compressible finite difference method are not well understood at present. Future work is needed to understand them.

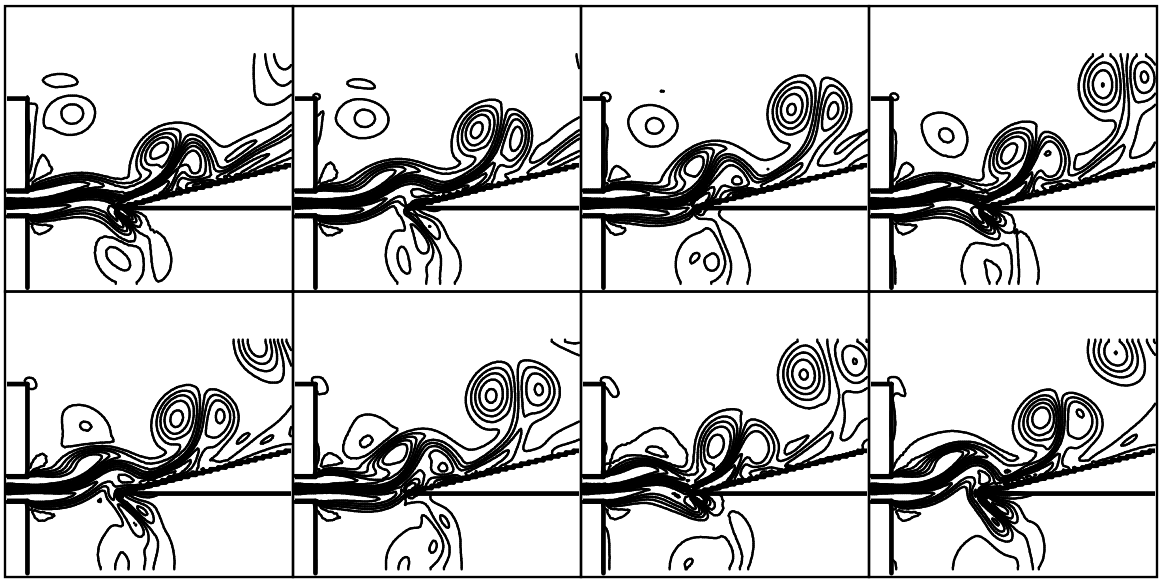

Figure 1-10: Jet oscillations in the flue-labium region. Frames are 0.33 ms apart, from left to right. Iso-vorticity contours are plotted.

To get an idea of how the jet of air moves inside the flue pipe, figures 1-9 to 1-13 show sequences of pictures of the flue-labium region from simulations using the lattice Boltzmann method. Similar pictures are obtained using the finite difference method. Figures 1-9, 1-10 come from a simulation of a closed-end soprano recorder which is 6.1 cm long and generates a tone of 1000 Hz (the blowing speed is 900 cm/s, and a complete picture of this recorder is shown in figure 6-1 of chapter 6). Figures 1-11 to 1-13 come from a simulation of a 20 cm closed-end recorder blown at 1104 cm/s. Figure 1-11 shows vorticity iso-contours, figure 1-12 shows the velocity vector field, and figure 1-13 shows kinetic energy iso-contours calculated as $V_1^2 + V_2^2$ and clipped between the values $1 - 2 \times 10^6$ $(\mathrm{cm/s})^2$.

Figure 1-9 illustrates the very beginning of blowing air into the recorder, and figures 1-10 to 1-13 illustrate the oscillations of the jet after startup. Initially, the jet of air turns outwards, and moves outside of the labium. This is simply because

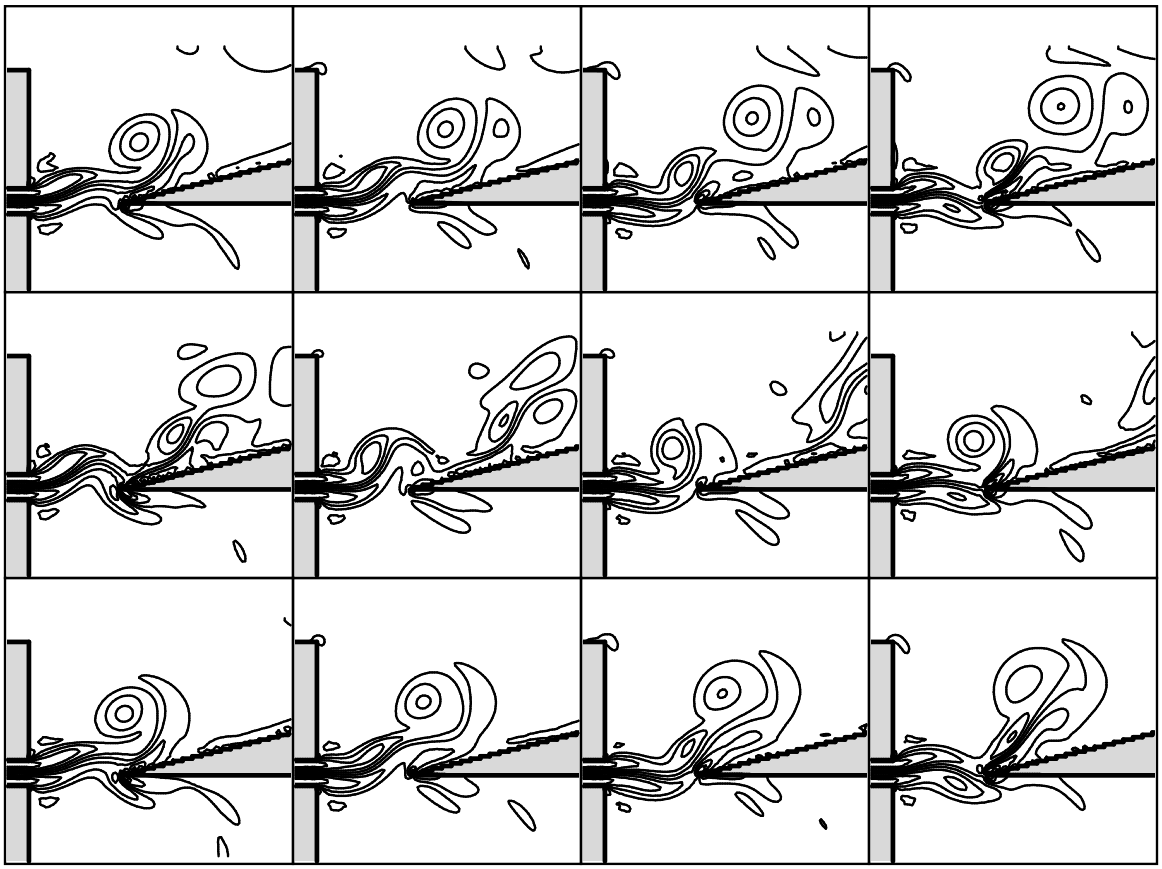

Figure 1-11: Jet oscillations of the 20 cm closed-end recorder at blowing speed 1104 cm/s. Frames are 0.22 ms apart, from left to right. Iso-vorticity contours are plotted. 35.6 ms after startup.

the pressure is smaller outside the pipe than inside. Subsequently, the jet begins to buckle, and starts to oscillate up and down. Meanwhile, the acoustic waves inside the pipe travel back and forth and build strong acoustic energy inside the pipe. The acoustic waves interact with the jet so that the jet oscillates at frequencies near the resonant frequencies of the pipe. Exactly how this happens is not known (section 7.2), but simple models have been proposed (Verge94 [57, 56], Hirschberg [26]). It would be an interesting future project to test these models against the precise data which can be obtained from the present simulations.

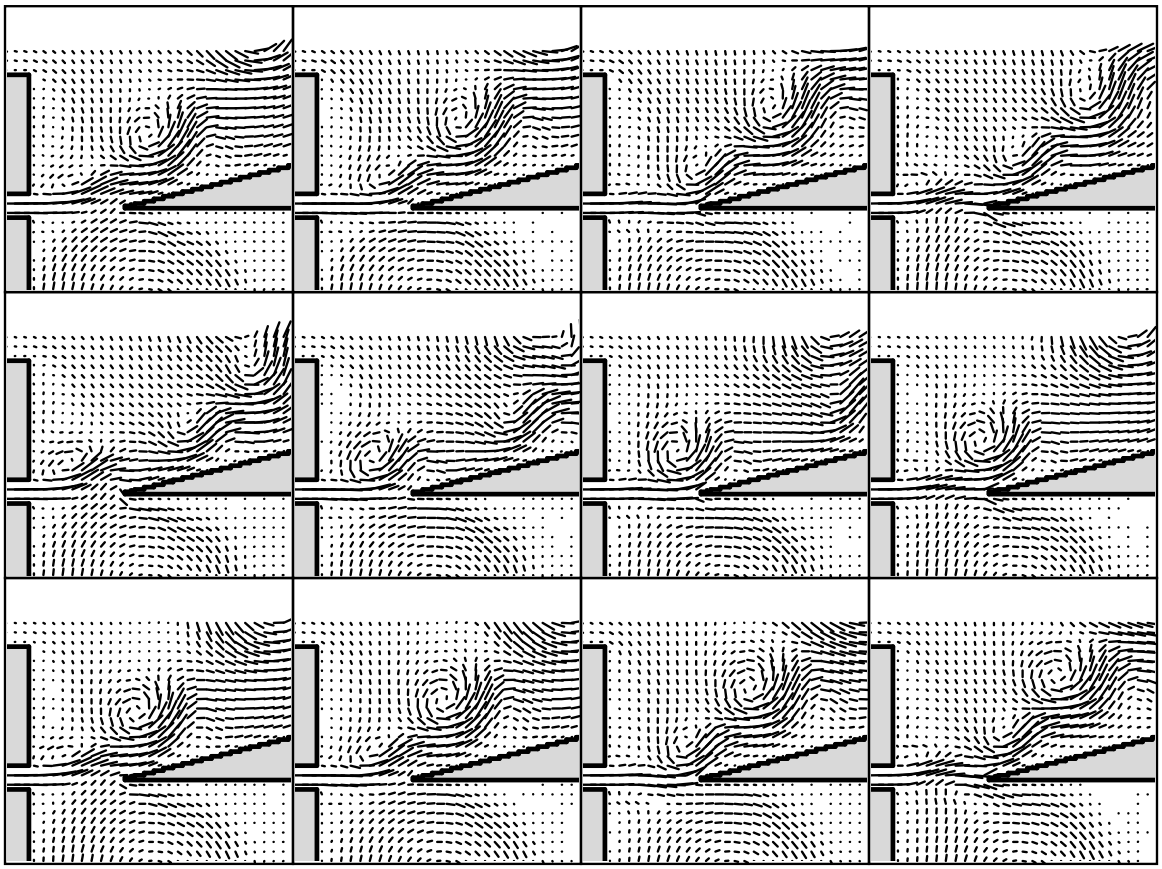

Figure 1-12: Jet oscillations of the 20 cm closed-end recorder at blowing speed 1104 cm/s. Frames are 0.22 ms apart, from left to right. The velocity vector field is plotted at 1 : 4 the actual grid resolution. 35.6 ms after startup.

1.4.3 Physical measurements

Comparing the simulations against physical measurements is very important because the physical measurements provide information of how close to reality the computer simulations are. Although the numerical accuracy of a numerical method can be tested on simple flow problems which possess exact solutions (this is done in chapter 4 for the lattice Boltzmann method), the numerical accuracy on simple problems does not guarantee that the modeling of a physical phenomenon is correct. There are many other factors that come into play when a real phenomenon is simulated. For instance,

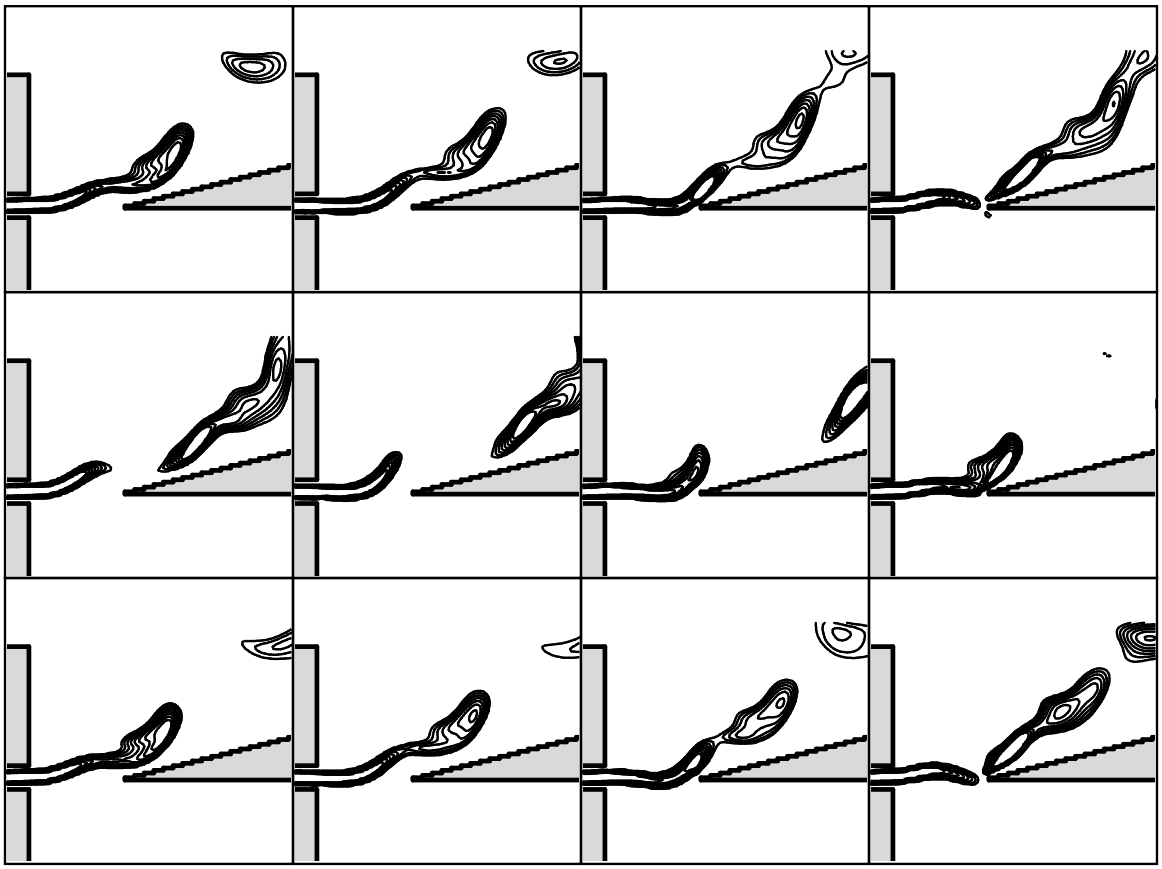

Figure 1-13: Jet oscillations of the 20 cm closed-end recorder at blowing speed 1104 cm/s. Frames are 0.22 ms apart, from left to right. Kinetic energy iso-contours are plotted. 35.6 ms after startup.

the underlying differential equations which are solved numerically (chapter 2) may miss some important effect of the physical phenomenon under consideration. Also, the numerical boundary conditions are often a poor model of the physical boundary conditions (for example, the practically-infinite outlet region above the recorder must be approximated with a small outlet region in the simulations). Thus, there is always some uncertainty about the physical modeling, which makes the comparison between simulations and physical measurements very important.

In the physical measurements presented in this section, a mechanical air supply

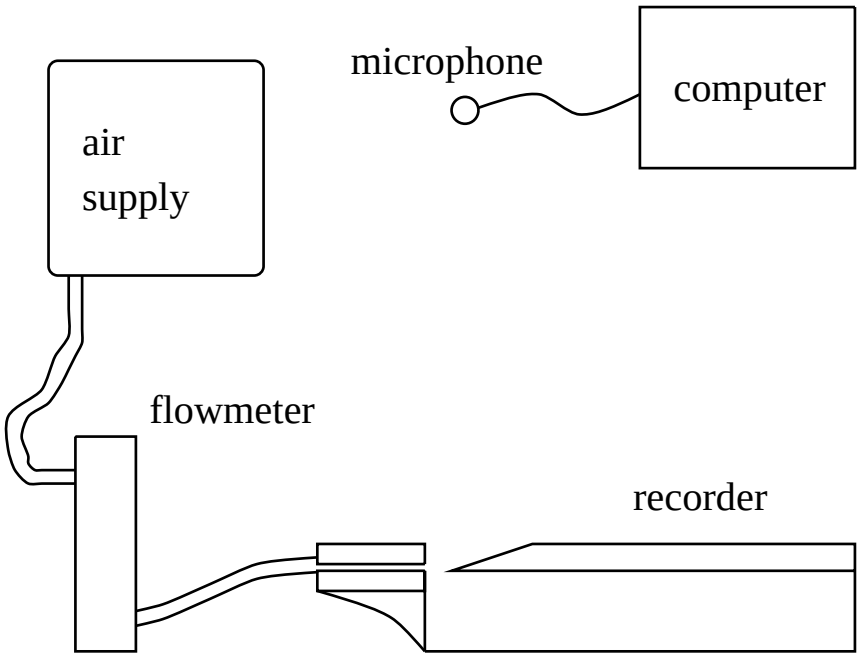

Figure 1-14: The setup for physical measurements. Not drawn to scale.

is used to blow air into the recorder. The air passes through a regulating valve and a flow-meter before reaching the recorder, as shown in figure 1-14. Thus, the response of the recorder can be measured for different blowing speeds. The generated acoustic signal is measured by means of a CT329 microphone, which is placed at a distance of approximately 100 cm away from the recorder. The analog signal from the microphone is digitized using a SONY portable computer with an internal A/D converter. Then, a Fourier transform is performed to calculate the frequency spectrum.

Figures 1-23 to 1-27 show acoustic signals obtained from physical measurements of the 20 cm closed-end recorder, and table 1.3 summarizes the frequencies. The acoustic signals are sampled during steady state (a few seconds after the initial blowing of air into the recorder). The sampling interval is 2.65×10^{-5} s, and corresponds to a maximum frequency of 18.9 kHz. The absolute amplitude of each measurement is not known because the measuring apparatus is not calibrated. However, the relative

amplitudes can be compared between different measurements because the measuring apparatus is identical in all cases.

A comparison between figures 1-23 to 1-27 shows that the amplitude of the acoustic signal increases with larger blowing velocity. Also, acoustic modes of higher frequency are excited as the blowing speed increases. It should be noted that a frequency of 1918 Hz (see table 1.3) is generated at the blowing speed of 2420 cm/s only when the initial blowing of air is abrupt. By contrast, a smooth (slow-rise) initial blowing of air makes the recorder generate the lower mode near 1145 Hz. Such behavior is expected in flue pipes (Verge94 [56]).

Another observation is that the frequencies generated by the recorder are related by ratios of integers such as $1 : 3 : 5 : 7 : 9$ which are characteristic of an open-closed pipe. For comparison purposes, table 1.4 shows the ideal resonant frequencies of an open-open and an open-closed pipe which is 20 cm long. The ideal resonant frequencies are based on the simple model of a pipe as a finite-length string with appropriate boundary conditions at the two ends. We can see that the ideal resonant frequencies of an open-closed pipe are similar to the frequencies generated by the flue, but there are differences. This is because the flue generates acoustic oscillations according to a complex nonlinear feedback between the acoustic waves in the pipe and the hydrodynamic behavior of the jet of air.

Finally, it must be noted that the blowing velocities of 1140 cm/s and 1558 cm/s produce a sound which includes a weak low-frequency beat (perhaps $10 - 20$ Hz). This beat is not visible in the frequency spectra shown in figures 1-24 and 1-25, but it can be clearly heard by the human ear. The low-frequency beat is an interesting issue to investigate in the future, but is not critical for an approximate comparison between the simulations and the physical measurements.

1.4.4 Comparison between simulation and measurements

Overall, the simulations are in reasonable agreement with the physical measurements. For instance, the lowest mode of 400 Hz, as well as the higher modes near 1200 Hz and 2000 Hz are predicted by the simulations. The qualitative behavior of jumping to higher modes with higher blowing speeds occurs both in the simulations and in the physical world. On the other hand, there are differences also.

The major difference (or cause of differences) between the simulations and the physical measurements is that the simulations correspond to the first 30-40 ms after startup, and the measurements correspond to the steady state a few seconds after startup (see figure 7-16 of section 7.5 for physical measurements of a startup transient). In this regard, only a rough comparison is possible between the simulations and the physical measurements. A rough comparison is possible because periodic oscillations become distinct 20 ms after startup, and the frequencies of the generated sound can be clearly observed.

It must be noted that computer simulations of the steady state (for example, one second after startup) would take a lot longer than the present simulations. Furthermore, a regular flow pattern exiting the outlet region would have to be established. To perform such simulations, improved boundary conditions are needed for the outlet region, as well as more compute-power, and perhaps a non-uniform grid to save on computational effort. Also, it should be noted that the startup transient is very sensitive to the details of the experimental apparatus. Thus, for the sake of simplicity, physical measurements of the steady state are considered here.

Leaving aside the issue of steady state versus initial response, it is worth noting that the acoustic signal is much cleaner (pure tones) in the physical measurements than in the simulations. [3] Also, the simulated recorder does not sing well at blowing

[3]The "dip" of the density signal in figure 1-17 at time 150×0.206 ms is caused by a very small vortex that reaches the sampling location, and subsequently moves away. Such a dip is expected because the density inside a vortex is much smaller than outside (tornado effect). Larger vortices have a much more pronounced effect than the one shown here. To avoid such effects, the acoustic

speed 818 cm/s, and the acoustic signal appears to die 20-30 ms after startup (this is discussed further in section 7.4). Specific modeling issues which may account for the above and other differences between the simulations and the physical measurements are as follows:

- The physical measurements sample the acoustic signal at 100 cm away from the recorder, while the simulations measure the acoustic signal 5 cm above the recorder.

- Three-dimensional effects are neglected in the simulations. It is possible that a 3D jet of air behaves slightly differently than a 2D jet. Also, a 3D resonant pipe can store more acoustic energy than a 2D resonant pipe. Thus, an exact correspondence between 2D and 3D at each blowing speed may not be possible.

- Higher spatial resolution than the one employed here ($\Delta x = 0.01$ cm) may be needed in the flue-labium region to follow the up/down motion of the jet, but perhaps not. A related issue is that the surface of the labium is rough at very small length scales $\Delta x = 0.01$ cm (see figures 1-8 and 1-7). The roughness of the labium may affect the shedding of vortices. However, it is probably a minor issue at the length scale of $\Delta x = 0.01$ cm, and it diminishes with smaller Δx.

- The walls of the outlet region near and above the labium reflect acoustic waves. Such walls are not present in the physical experiments. It is possible that the reflections from the walls influence the operation of the flue. However, I expect that the effect is small because the very-top boundary of the outlet region does not reflect acoustic waves (where the flow exits from the simulation).

- The walls of the outlet region may affect the buildup of hydrodynamic pressure gradients above the flue. The operation of the flue is very sensitive to the surrounding pressure gradients.

---

signal should *not* be sampled very near and above the labium where vorticity is shed.

- The limited size and the two-dimensional form of the outlet region encourage the accumulation of vortices right above the labium. The vortices introduce hydrodynamic pressure gradients, and may interfere with the oscillations of the jet. By contrast, in the physical world (practically infinite and three-dimensional) the generated vorticity is quickly carried away from the sensitive region of the flue and labium. In the simulations, the vorticity can not move away so easily.

Anyone of the above issues, or a combination of them may be responsible for the differences between the simulations and the physical measurements. However, the most important issue seems to be the modeling of the outlet region. Future work should be done along the following directions:

- Improve the boundary conditions at the outlet.
- Devise suitable means of clearing the outlet region from accumulated vorticity.
- Employ non-uniform grid to enlarge the outlet region without incurring a large computational cost.

Despite the differences between the simulations and the physical measurements, the results are very good as a first step.

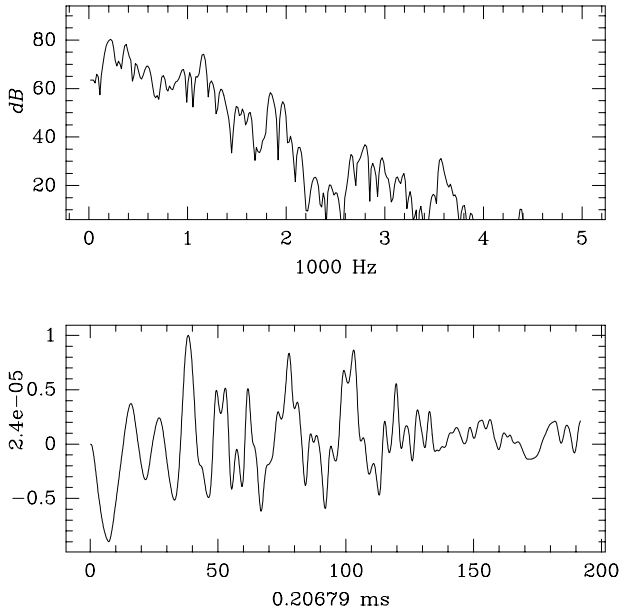

Figure 1-15: Lattice Boltzmann method, 20 cm closed-end soprano recorder, blowing velocity 818 cm/s.

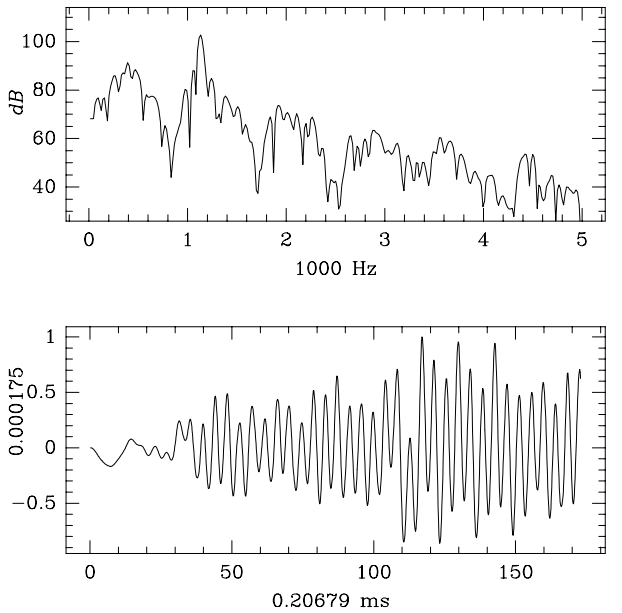

Figure 1-16: Lattice Boltzmann method, 20 cm closed-end soprano recorder, blowing velocity 1104 cm/s.

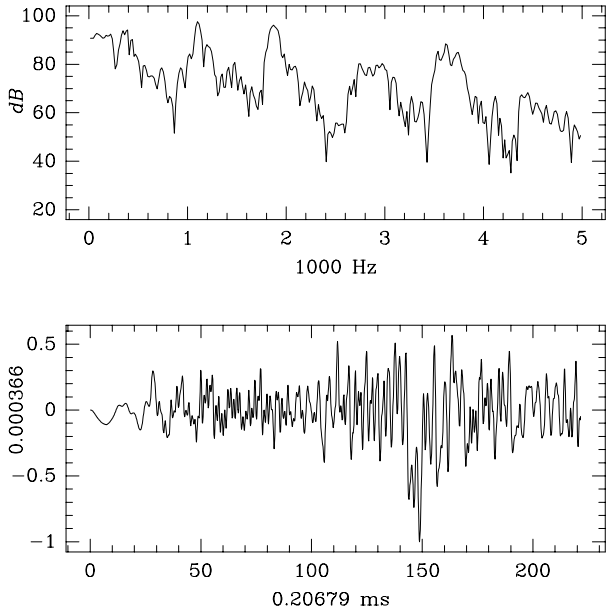

Figure 1-17: Lattice Boltzmann method, 20 cm closed-end soprano recorder, blowing velocity 1535 cm/s.

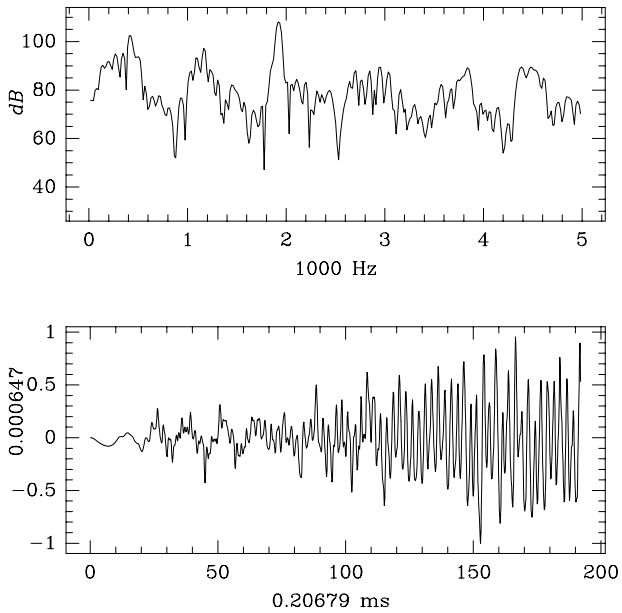

Figure 1-18: Lattice Boltzmann method, 20 cm closed-end soprano recorder, blowing velocity 1995 cm/s.

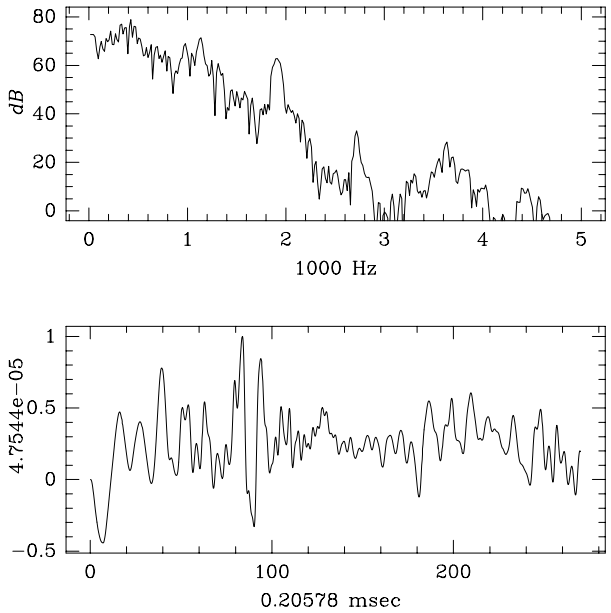

Figure 1-19: Compressible finite difference method, 20 cm closed-end soprano recorder, blowing velocity 838 cm/s.

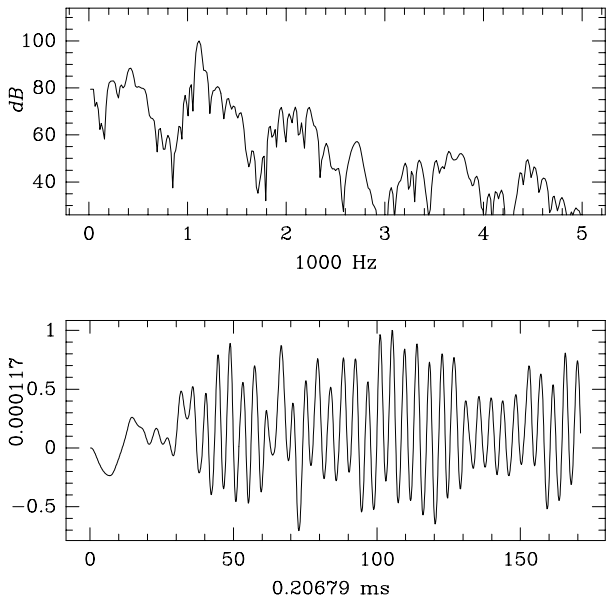

Figure 1-20: Compressible finite difference method, 20 cm closed-end soprano recorder, blowing velocity 1113 cm/s.

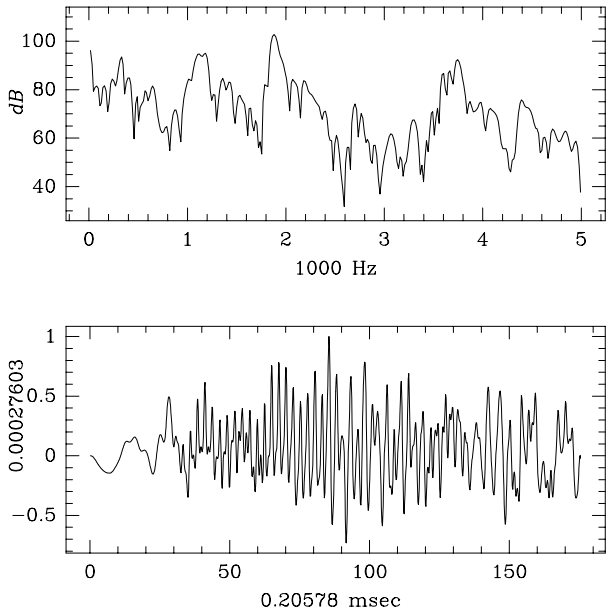

Figure 1-21: Compressible finite difference method, 20 cm closed-end soprano recorder, blowing velocity 1634 cm/s.

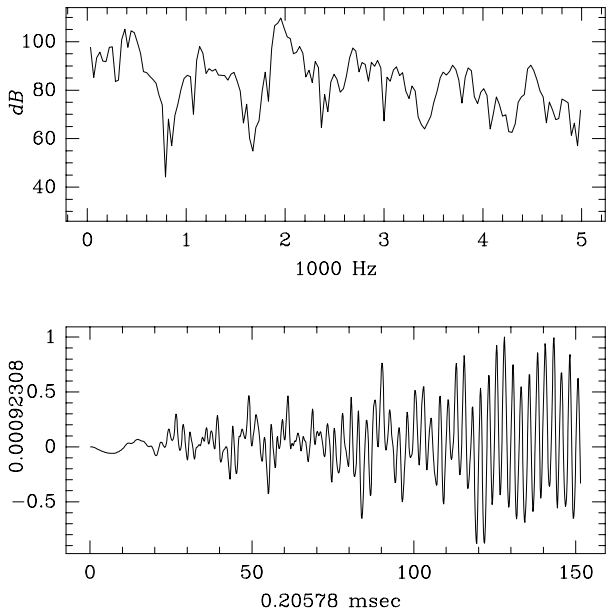

Figure 1-22: Compressible finite difference method, 20 cm closed-end soprano recorder, blowing velocity 2082 cm/s.

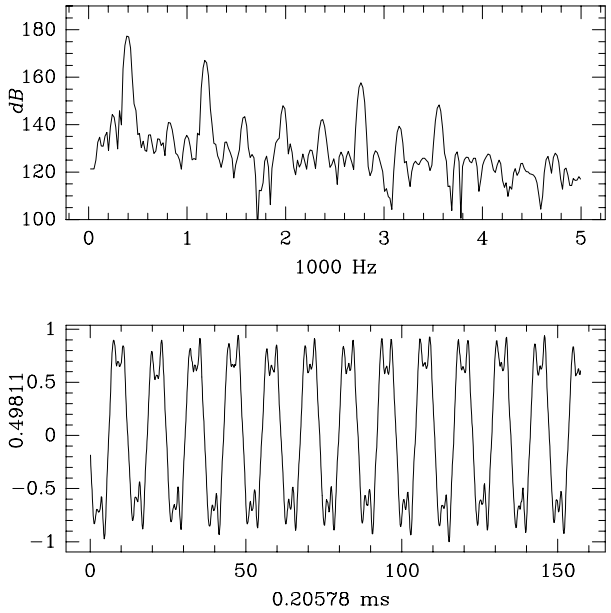

Figure 1-23: Physical measurements, steady state, 20 cm closed-end soprano recorder, blowing velocity 734 cm/s. Arbitrary units of amplitude.

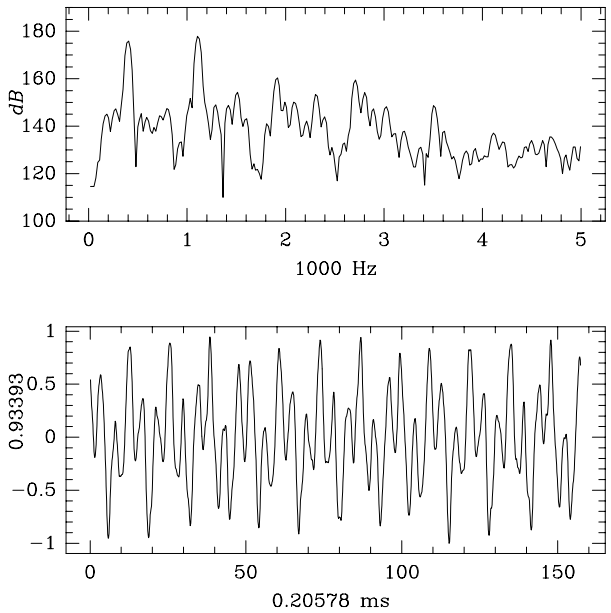

Figure 1-24: Physical measurements, steady state, 20 cm closed-end soprano recorder, blowing velocity 1140 cm/s. Arbitrary units of amplitude.

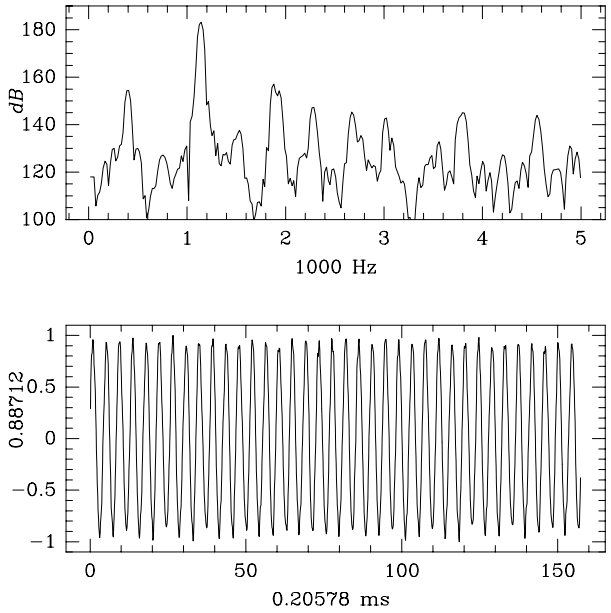

Figure 1-25: Physical measurements, steady state, 20 cm closed-end soprano recorder, blowing velocity 1558 cm/s. Arbitrary units of amplitude.

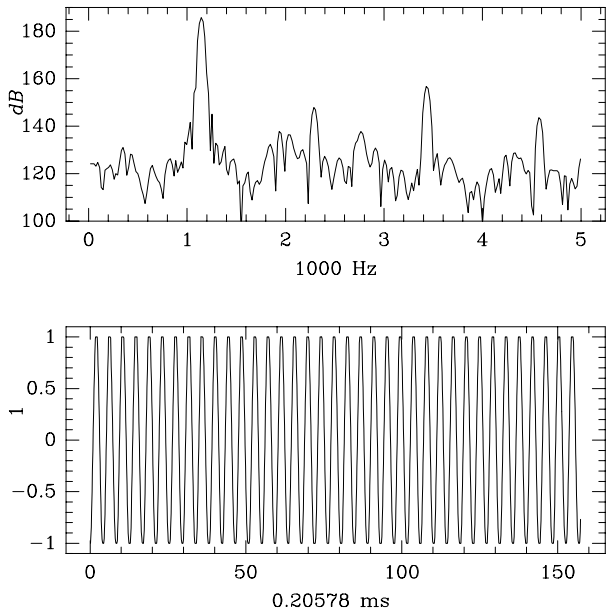

Figure 1-26: Physical measurements, steady state, 20 cm closed-end soprano recorder, blowing velocity 1985 cm/s. Arbitrary units of amplitude.

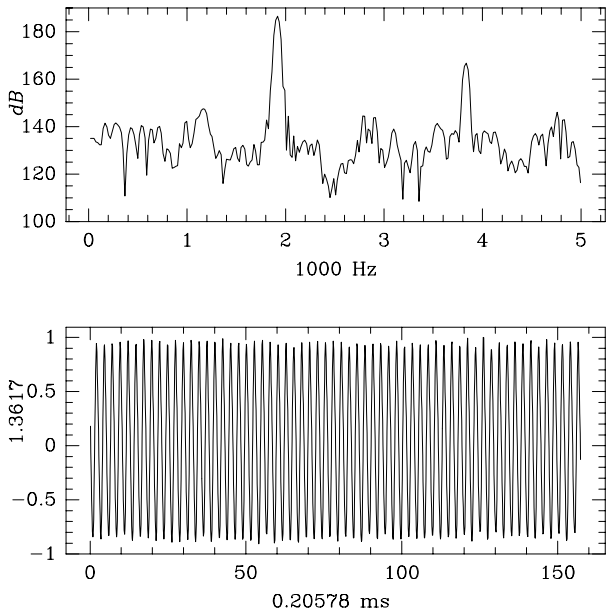

Figure 1-27: Physical measurements, steady state, 20 cm closed-end soprano recorder, blowing velocity 2420 cm/s, and abrupt blow of air at startup. Arbitrary units of amplitude.

Chapter 2

The motion of fluids

In this chapter, the partial differential equations of fluid flow, known as the Navier Stokes equations, are derived in the context of phenomena such as the flow of air at room temperature and atmospheric pressure. In addition, an introduction to hydrodynamics and acoustics is presented which is useful background material. Most of the results of this chapter are not really new as one can infer from the references to previous work. However, the results are re-derived here and presented in a novel way with extra care to be correct and relevant to physical reality. In addition, some discussions such as the paradox of incompressibility in section 2.4.3 and the justification of omitting the bulk viscosity in subsonic flow, can not be found easily in the literature as far as I know.

2.1 The scale of macroscopic flow

A fluid can be modeled either at the microscopic level or at the macroscopic level. Here, the flow of a fluid is modeled at the macroscopic level where "macroscopic" means that the fluid is viewed as a continuum and that the underlying molecular motion is not considered directly. In particular, it is assumed that an infinitesimal volume of fluid can be defined which is very large compared to the microscopic scales of

molecular motion, and simultaneously very small compared to the macroscopic scales of fluid flow (Batchelor [3, p.4] and Tritton [54, p.48]). Thus, microscopic statistical fluctuations are ignored, and the state of the fluid is defined as a continuous function of space and time.

The above discussion can be made more precise by considering some numbers. The diameter of an air molecule (modeled as a hard core sphere or billiard ball) is of the order 3×10^{-8} cm (Batchelor [3, p.3], Skordos&Zurek [49, p.878]). The mean free path (average distance traveled by a molecule between collisions) is of the order 10^{-5} cm at room temperature and atmospheric pressure. The smallest length scale where the macroscopic fluid dynamics can be safely employed is about 10^{-3} cm, namely, 100 times the mean free path. Occasionally, macroscopic fluid dynamics (the Navier Stokes equations) are employed at length scales as small as the mean free path, for example, in ultrasonic acoustics (Morse&Ingard [33]). However, there is no reason to consider such small length scales here, and 10^{-3} cm will be assumed to be the smallest length scale of interest. It should be noted that an acoustic wavelength of 10^{-3} cm corresponds to an acoustic frequency of 34 MHz.

2.2 The conservation laws

The three most important properties of fluid flow are the conservation of mass, momentum, and energy. These conservation properties arise from the underlying molecular dynamics of fluids, and they are inherited by the macroscopic dynamics. The conservation properties are so powerful that one can derive the Navier Stokes equations by imposing conservation at the microscopic level, and by performing macroscopic averaging of the microscopic dynamics (Huang [27]). Such a derivation is called the kinetic theory approach. A simplified version of kinetic theory can be found in section 4.1.2, where it is shown that the lattice Boltzmann method approximates the Navier Stokes equations through a kinetic theory expansion known as the Chapman-

Enskog expansion.

Besides the kinetic theory approach, another way of deriving the Navier Stokes equations is to assume that the conservation of mass, momentum, and energy apply directly at the macroscopic level. Specifically, an infinitesimal but macroscopic volume of fluid (called a fluid element) is considered, and its evolution in time is examined. The mass of the fluid element must remain constant as the fluid element moves with the flow. The momentum and energy may change as a result of interactions with the surrounding fluid elements, but the interactions must conserve the total momentum and energy. By considering small changes during a sufficiently small interval of time, a set of partial differential equations can be derived which describe the evolution of mass, momentum, and energy of individual fluid elements.

An important simplification in deriving the macroscopic equations of fluid flow is to introduce flow variables (density, velocity, and temperature) which are functions of space and time. The flow variables are an alternative way of describing the flow as opposed to the mass, momentum, and energy of individual fluid elements . The two approaches are equivalent. For instance, by integrating the values of the flow density and velocity inside a given volume of space at a particular point in time, we can obtain the mass and the momentum of a fluid element that corresponds to the volume of space under consideration at that particular time.

The flow variables are simpler to use than the mass, momentum, and energy of individual fluid elements because the flow variables are defined on a fixed coordinate system, and do not move with the flow as the fluid elements do (Morse&Ingard [33, p.235], Batchelor [3, p.71], Lamb [31, p.12]). When the description of a flow is based on the flow variables only, it is called Eulerian. Alternatively, when the description of a flow refers to the properties of individual fluid elements, it is called Lagrangian. Most texts in fluid mechanics follow the Eulerian description, and this will be done here also.

Below, the Navier Stokes equations are derived using the ideas outlined above. For

this purpose, the fluid density $\rho(x, y, z, t)$ and the fluid velocity $V_j(x, y, z, t)$ are introduced as continuous functions of space and time, where the components of the fluid velocity V_j correspond to the Cartesian directions x, y, z for $j = 1, 2, 3$ respectively. Also, the advective derivative D/Dt is introduced as follows,

$$\frac{D}{Dt} \;=\; \frac{\partial}{\partial t} \;+\; V_j \, \frac{\partial}{\partial x_j} \tag{2.1}$$

where the Einstein summation convention is used: When an index appears twice in the same term, a summation is automatically implied. The notation x_j stands for x, y, z when $j = 1, 2, 3$. The advective derivative is a special case of the total derivative of a variable which is a function of x, y, z, t under the following assumption,

$$\frac{\partial x_j}{\partial t} \;=\; V_j \tag{2.2}$$

The above assumption is true in the case of a fluid element which moves with the local velocity V_j of the flow. It turns out that the advective derivative is omnipresent in fluid mechanics, and it is worth reserving the symbol D/Dt to refer to the advective derivative (Batchelor [3, p.73]).

2.2.1 Mass conservation

First, the mass conservation equation is derived, which is also known as the mass continuity equation. We consider a fluid element which is positioned at x, y, z at time t, and has volume $A(x, y, z, t)$. The mass of the fluid element is conserved and is equal to ρA. Therefore, the total derivative of the mass must be zero, or actually the advective derivative $D/\Delta t$ must be zero because the fluid element moves with the local velocity V_j of the flow. Thus,

$$\frac{D}{Dt}\,(\rho A) \;=\; 0 \tag{2.3}$$

which gives,

$$A\frac{D\rho}{Dt} + \rho\frac{DA}{Dt} \;=\; 0 \tag{2.4}$$

and

$$\frac{D\rho}{Dt} + \rho \left(\frac{1}{A} \frac{DA}{Dt} \right) = 0 \tag{2.5}$$

To proceed further, we need to express the relative change of the volume of the fluid element $(1/A\ DA/Dt)$ in terms of the flow variables. As we will see below, the relative change of the volume of the fluid element (also known as *dilatation*) is equal to the divergence of the fluid velocity,

$$\left(\frac{1}{A} \frac{DA}{Dt} \right) = \frac{\partial V_j}{\partial x_j} \tag{2.6}$$

To prove equation 2.6, we examine how the geometry of the fluid element distorts as the fluid element moves with the flow. Following Lamb [31, p.5], we consider a cubic fluid volume such as the one shown in figure 2-1. We assume that the six faces of the cubic volume are initially aligned with the axes of the coordinate system. The center of the volume is located at some point (x_1, x_2, x_3), and the volume has dimensions $(\Delta x_1, \Delta x_2, \Delta x_3)$. The two faces of the cube that are opposite each other along the x_1 direction are referred to as the x_1-faces of the cube, and they are located at

$$(x_1 \pm \frac{\Delta x_1}{2}, x_2, x_3) \tag{2.7}$$

If the fluid velocity is equal to (V_1, V_2, V_3) at the center point (x_1, x_2, x_3) of the cube, then the x_1-faces are moving outwards (expanding) with the following velocities along the x_1 direction,

$$\left(V_1 + \frac{\partial V_1}{\partial x_1} \frac{\Delta x_1}{2} \right) \tag{2.8}$$

$$- \left(V_1 - \frac{\partial V_1}{\partial x_1} \frac{\Delta x_1}{2} \right) \tag{2.9}$$

The above quantities express the change of volume along the x_1 direction. The motion of the x_1-faces along the x_2 and x_3 directions produces shearing of the volume only, and does not change the volume to first order in the differential quantities $\Delta x_1, \Delta x_2, \Delta x_3$. Thus, we can ignore the shearing motion here. After an infinitesimal

interval of time Δt has elapsed, the change of volume due to expansion along the x_1 direction is equal to

$$\left(\frac{\partial V_1}{\partial x_1}\,\Delta x_1\right)\Delta x_2 \Delta x_3 \Delta t \tag{2.10}$$

Similar relations can be obtained for the expansion along the x_2,x_3 directions using the other faces of the cubic volume. The total rate of change of volume per unit of time (expanding volume) is given by the sum of the above terms,

$$\frac{DA}{Dt} = \left(\frac{\partial V_1}{\partial x_1} + \frac{\partial V_2}{\partial x_2} + \frac{\partial V_3}{\partial x_3}\right)\Delta x_1 \Delta x_2 \Delta x_3 \tag{2.11}$$

Combining equation 2.11 with equation 2.5 and the fact that $A = \Delta x_1 \Delta x_2 \Delta x_3$ we obtain,

$$\frac{D\rho}{Dt} + \rho\frac{\partial V_j}{\partial x_j} = 0 \tag{2.12}$$

where the summation convention is used. We also use the notation,

$$\frac{D\rho}{Dt} + \rho(\nabla \cdot \vec{V}) = 0 \tag{2.13}$$

The above is the mass continuity equation. We have derived it by considering the conservation of mass of a moving fluid element during an infinitesimal interval of time, and by relating the mass of the fluid element to the Eulerian density and velocity of the flow.

An alternative way of deriving the mass continuity equation is to consider a fixed-in-space volume of fluid, and to balance the mass which flows through the boundaries of the volume with the change of density inside the volume. This alternative approach is found in Landau&Lifshitz [32, p.1] and Batchelor [3, p.74], and it produces the following equation,

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{V}) = 0 \tag{2.14}$$

which is equivalent to equation 2.13. In my opinion, the approach of the moving fluid volume is somewhat more intuitive than the fixed-in-space volume because it is easier to visualize what happens when the fluid volume moves and distorts with the flow. On the other hand, the use of both approaches leads to a better understanding

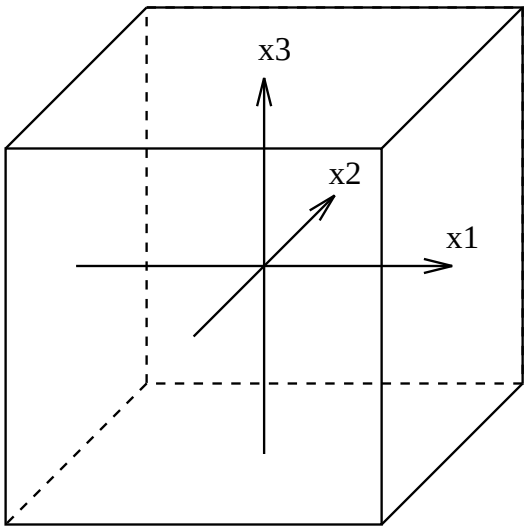

Figure 2-1: A fluid element whose shape is a cube at time zero, and its six faces are normal to the Cartesian axes.

than either one by itself. It should be noted that the fixed-in-space volume is a purely Eulerian approach; while the moving fluid volume, as described above, is a Lagrangian idea expressed in Eulerian flow variables.

2.2.2 Momentum conservation

The momentum Navier Stokes equation can be derived in a similar way to the mass conservation equation by considering the changes of momentum of a fluid element during an infinitesimal interval of time. If we consider the forces acting on the six faces of a cubic volume, we can write an equation for the conservation of momentum along the x_j direction, as follows,

$$\frac{D(\rho V_j)}{Dt} = \frac{\partial(\sigma_{jk})}{\partial x_k} \tag{2.15}$$

where σ_{jk} is called the *pressure tensor*, and it models the forces that arise from pressure and from viscosity (internal friction of the fluid medium). The derivation of the pressure tensor is somewhat long and is omitted here. The details can be found in standard textbooks such as Landau&Lifshitz [32, p.45], Batchelor [3, p.147],

Newman [34, pp.50–63]. These references show that the pressure tensor can be written as follows,

$$\sigma_{jk} = -P\delta_{jk} \; + \; \eta \left(\frac{\partial V_j}{\partial x_k} + \frac{\partial V_k}{\partial x_j} \right) \; + \; (-\frac{2}{3}\eta \; + \; \zeta) \, (\nabla \cdot \vec{V}) \, \delta_{jk} \tag{2.16}$$

where P is the scalar pressure, η is the first coefficient of viscosity (corresponding to friction from shearing motion), and ζ is the second coefficient of viscosity (corresponding to friction from bulk-expanding motion). The above tensors can be represented in a Cartesian coordinate system in terms of 3×3 matrices as follows,

$$\delta_{jk} = \left\{ \begin{array}{ccc} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array} \right\} \tag{2.17}$$

$$\left(\frac{\partial V_j}{\partial x_k} + \frac{\partial V_k}{\partial x_j} \right) = \left\{ \begin{array}{ccc} 2\frac{\partial V_1}{\partial x} & \frac{\partial V_1}{\partial y} + \frac{\partial V_2}{\partial x} & \frac{\partial V_1}{\partial z} + \frac{\partial V_3}{\partial x} \\ \frac{\partial V_2}{\partial x} + \frac{\partial V_1}{\partial y} & 2\frac{\partial V_2}{\partial y} & \frac{\partial V_2}{\partial z} + \frac{\partial V_3}{\partial y} \\ \frac{\partial V_3}{\partial x} + \frac{\partial V_1}{\partial z} & \frac{\partial V_3}{\partial y} + \frac{\partial V_2}{\partial z} & 2\frac{\partial V_3}{\partial z} \end{array} \right\} \tag{2.18}$$

Further, the following identities are useful,

$$\frac{\partial}{\partial x_k}(P\delta_{jk}) = \frac{\partial P}{\partial x_j} \tag{2.19}$$

and

$$\begin{aligned} \frac{\partial}{\partial x_k} \left(\frac{\partial V_j}{\partial x_k} + \frac{\partial V_k}{\partial x_j} \right) \; &= \; \left(\frac{\partial V_j}{\partial x_k \partial x_k} \right) \; + \; \frac{\partial}{\partial x_j} \left(\frac{\partial V_k}{\partial x_k} \right) \\ &= \quad \nabla^2 V_j \quad + \; \frac{\partial}{\partial x_j}(\nabla \cdot \vec{V}) \end{aligned} \tag{2.20}$$

The above identities can be used to write the momentum Navier Stokes equations in the following form,

$$\frac{D(\rho V_j)}{Dt} = \; -\frac{\partial P}{\partial x_j} \; + \; \eta \nabla^2 V_j \; + \; \left(\frac{\eta}{3} + \zeta \right) \frac{\partial}{\partial x_j}(\nabla \cdot \vec{V}) \tag{2.21}$$

where $j = 1, 2, 3$. The physical interpretation of the various terms of the above momentum equation will be discussed in section 2.4. Next, the conservation of energy is examined.

2.3 Adiabatic variations of temperature

In general, the simulation of viscous flow with acoustic waves requires the complete Navier Stokes equations: the mass continuity equation, three equations for momentum conservation, an equation for energy conservation, and an equation of state which relates the three thermodynamic variables (temperature, pressure, and density). The temperature represents the internal energy of the fluid, and arises from the internal degrees of freedom such as the vibrations of the fluid molecules. The energy equation couples together the temperature variations with the density and momentum variations of the flow.

In special cases, such as the flow of air at room temperature and atmospheric pressure, the coupling between the temperature and the momentum of the flow is very small and can be neglected. In particular, the partial differential equation for energy conservation can be replaced with an exact relation between the temperature, density, and pressure. This relation is called the adiabatic approximation, and it is employed in the simulations presented here, in order to avoid solving a partial differential equation corresponding to the conservation of energy.

In the adiabatic approximation, it is assumed that there is no conduction of heat between different parts of the flow. In addition, it is assumed that there are local heat reservoirs at each point in space which allow local temperature oscillations, but without any conduction of heat. The local heat reservoirs are necessary because the density fluctuations of acoustic waves are accompanied by small, but non-negligible temperature fluctuations. Namely, when the air suddenly compresses, its temperature rises; when the air expands, its temperature lowers.

The justification for the adiabatic approximation is easy to understand in the case acoustic oscillations: the acoustic oscillations happen very fast, so it makes sense to assume that there is no conduction of heat. However, the adiabatic approximation applies more generally as we shall see afterwards. First, let us derive an exact relation between the temperature, density, and pressure by considering the temperature fluctuations of acoustic waves. This idea is due to Laplace, and is explained very nicely in Rayleigh's book [42, p.20]. Mathematically, we define P_0, ρ_0, θ_0 the initial values of pressure, density, and temperature, and P, ρ, θ the new values after an adiabatic change. Then, the following relation applies which is known as the adiabatic law (see section 2.3.1 for a derivation),

$$\frac{P}{P_0} = \left(\frac{\rho}{\rho_0}\right)^{\gamma} = \left(\frac{\theta}{\theta_0}\right)^{\left(\frac{\gamma}{\gamma-1}\right)} \tag{2.22}$$

where γ is the ratio of the specific heats of the gas, and it is equal to 1.4 in the case of air. We also define the small variations P', ρ', θ' around the constant mean values P_0, ρ_0, θ_0 as follows,

$$\begin{aligned} P &= P_0 + P' \\ \rho &= \rho_0 + \rho' \\ \theta &= \theta_0 + \theta' \end{aligned} \tag{2.23}$$

We can obtain a relation between the variations P' and θ' by expanding the following sum to first order in small quantities,

$$\frac{P}{P_0} = 1 + \frac{P'}{P_0} = \left(1 + \frac{\rho'}{\rho_0}\right)^{\gamma} \simeq 1 + \gamma\frac{\rho'}{\rho_0} \tag{2.24}$$

Therefore,

$$P' = \left(\gamma\frac{P_0}{\rho_0}\right)\rho' \tag{2.25}$$

To proceed further, we use the equation of state for gases, which is a relation between the mean values of the thermodynamic variables,

$$P_0 = R\,\rho_0\,\theta_0 \tag{2.26}$$

where θ_0 is expressed in absolute degrees Kelvin, and R is a gas constant which is equal to 2.870×10^6 cm^2/s^2 per degree Kelvin in the case of air (Batchelor [3, p.43] and Lamb [31, p.478]). Equations 2.25 and 2.26 give

$$P' = (\gamma R \theta_0) \rho' \tag{2.27}$$

which can be written as

$$P' = c_s^2 \rho' \tag{2.28}$$

with the definition

$$c_s = \sqrt{\gamma R \theta_0} \tag{2.29}$$

The constant c_s is the speed of the propagation of acoustic waves as we will see in section 2.5. The precise relation between pressure and density is as follows,

$$P = c_s^2 \rho + (P_0 - c_s^2 \rho_0) \tag{2.30}$$

For the sake of simplicity, the following formula is commonly used (throughout this work and elsewhere),

$$P = c_s^2 \rho \tag{2.31}$$

with the understanding that it is okay to subtract an arbitrary offset from the pressure because only the gradients of the pressure influence the flow.

Above, an exact relation between the density and the pressure has been derived by examining the adiabatic changes of pressure, density, and temperature of acoustic waves. It turns out that the adiabatic approximation applies more generally to any variations of density in subsonic flow as long as the variations are small. The reason is as follows. Let us consider a steady flow inside a pipe (Hagen-Poiseuille flow, Landau&Lifshitz [32, p.51]), and let us ask whether the relation between pressure and density variations $P = c_s^2 \rho$ still applies. The answer is yes, and the adiabatic law still applies because what is important is how the state of equilibrium is reached. Any disturbance in the fluid is transmitted by fast acoustic waves, so that the new

state of equilibrium is reached quickly and adiabatically to a good approximation. Accordingly, the relation between small variations of pressure and density which is derived above applies in general for any variations of density in subsonic flow.

For historical interest, it should be noted that Laplace proposed the adiabatic law between pressure and density in order to calculate the speed of sound using equation 2.29. Before Laplace's formula, the previous estimate of the speed of sound fell short of experimental measurements. The previous estimate, attributed to Newton, assumed Boyle's law of infinitely slow changes at constant temperature,

$$\frac{P}{P_0} = \frac{\rho}{\rho_0} \tag{2.32}$$

which misses the constant factor γ so that the speed of sound comes out short by a factor $\sqrt{\gamma} = 1.18$.

2.3.1 Derivation of the adiabatic law

For completeness, a derivation of the adiabatic law (equation 2.22) is presented here, which follows closely the derivation of Rayleigh [42, p.21]. First, the equation of state for gases is considered which relates the three thermodynamic variables pressure P, density ρ, and temperature θ,

$$P = \rho\, R\, \theta \tag{2.33}$$

This can also be written as,

$$P\, A = R'\, \theta \tag{2.34}$$

where A is the volume under consideration, and is related to the density ρ as follows,

$$\frac{dA}{A} = -\frac{d\rho}{\rho} \tag{2.35}$$

The new gas constant R' is equal to the original gas constant R times the mass of the volume under consideration. Differentiation of equation 2.34 produces the differential equation of state which will be used below.

$$\frac{dP}{P} + \frac{dA}{A} = \frac{d\theta}{\theta} \tag{2.36}$$

First, it is noted that the equation of state constraints the three thermodynamic variables P, A, θ, any two of them can be taken as independent variables, for example P and A. Further, if an additional constraint is introduced, it may be possible to obtain an exact relation between the thermodynamic variables because only one variable will then be independent. The additional assumption is that there is no communication of heat in the medium.

In order to exploit the assumption of no communication of heat, we examine the amount of heat in the fluid volume as a function of the pressure P and the volume A. If the amount of heat is denoted Q, the following total differential expresses the conduction of heat in terms of changes in pressure and volume,

$$dQ \;=\; \left(\frac{dQ}{dA}\right) dA + \left(\frac{dQ}{dP}\right) dP \tag{2.37}$$

The above equation can be simplified by considering changes of heat under constant pressure, $dP = 0$, and also changes of heat under constant volume, $dA = 0$. In particular, using the differential equation 2.36, the following relations can be obtained,

$$\kappa_p \;=\; \left(\frac{dQ}{d\theta}\right)_P \;=\; \left(\frac{dQ}{dA}\right)\left(\frac{dA}{d\theta}\right) = \left(\frac{dQ}{dA}\right)\frac{A}{\theta} \tag{2.38}$$

$$\kappa_v \;=\; \left(\frac{dQ}{d\theta}\right)_A \;=\; \left(\frac{dQ}{dP}\right)\left(\frac{dP}{d\theta}\right) = \left(\frac{dQ}{dP}\right)\frac{P}{\theta} \tag{2.39}$$

The above quantities are the ratios of changes in heat divided by the changes in temperature under constant pressure and under constant volume. They are called specific heats, and they are constant within a wide range of temperatures and pressures (Batchelor [3, p.44]). They are certainly constant for the purpose of modeling air flow inside flue pipes. Using the above relations together with the assumption that there is no conduction of heat, $dQ = 0$, equation 2.37 becomes,

$$dQ \;=\; \left(\kappa_p \frac{\theta}{A}\right) dA \;+\; \left(\kappa_v \frac{\theta}{P}\right) dP \;=\; 0 \tag{2.40}$$

or

$$-\,\kappa_p \frac{d\rho}{\rho} \;+\; \kappa_v \frac{dP}{P} \;=\; 0 \tag{2.41}$$

which gives

$$\frac{dP}{d\rho} = \left(\frac{\kappa_p}{\kappa_v}\right)\frac{P}{\rho} \tag{2.42}$$

or equivalently,

$$\frac{dP}{d\rho} = \gamma\frac{P}{\rho} \tag{2.43}$$

with the appropriate definition of the constant γ,

$$\gamma = \kappa_p/\kappa_v \tag{2.44}$$

By performing an integration of equation 2.43 using logarithms, the adiabatic law is obtained,

$$\frac{P}{P_0} = \left(\frac{\rho}{\rho_0}\right)^{\gamma} \tag{2.45}$$

where P_0, θ_0 are two initial values. This is the adiabatic law of equation 2.22 which we wanted to prove.

We recall that the adiabatic law of equations 2.22 and 2.45 was the starting point for calculating the relation between small variations of pressure and density $P' = c_s^2\rho'$. Here, we can also see that an alternative way of deriving the relation $P' = c_s^2\rho'$ would be to assume small variations P', ρ' around an initial point P_0, ρ_0 in equation 2.43 and write,

$$\frac{dP}{d\rho} = \gamma\frac{P}{\rho} \simeq \gamma\frac{P_0}{\rho_0} \tag{2.46}$$

An integration that involves the small variations P', ρ' gives

$$P' = \left(\gamma\frac{P_0}{\rho_0}\right)\rho' \tag{2.47}$$

This is the same relation between small variations of pressure and density which was derived previously in equation 2.25.

Armed with an exact relation between the pressure and the density, we can proceed in the following sections to analyze the physical properties of the Navier Stokes equations.

2.4 The Navier Stokes equations

The Navier Stokes equations which I use to model air flow inside flue pipes, can be written compactly as follows,

$$\frac{D\rho}{Dt} + \rho\nabla\cdot\vec{V} = 0 \tag{2.48}$$

$$\frac{D(\rho V_j)}{Dt} + c_s^2\frac{\partial\rho}{\partial x_j} - \nu\rho\,\nabla^2 V_j - \mu\rho\frac{\partial(\nabla\cdot\vec{V})}{\partial x_j} = 0 \tag{2.49}$$

They are partial differential equations which express the conservation of mass and momentum, and must be solved numerically. In addition, there is an exact relation relation between the temperature, the density, and the pressure according to the adiabatic law. This relation completes the physical model, and replaces a partial differential equation for energy conservation as explained in the previous section. The adiabatic relation between pressure and density variations is as follows,

$$P \;=\; c_s^2\,\rho \tag{2.50}$$

Regarding notation, the index j in the Navier Stokes equations runs between $j = 1, 2, 3$. The symbol D/Dt is the advective derivative, and c_s is the speed of sound. The coefficients ν and μ are density-normalized viscosity coefficients which are defined as follows,

$$\nu \;=\; \frac{\eta}{\rho} \qquad \mu \;=\; \frac{\eta/3 \,+\, \zeta}{\rho} \tag{2.51}$$

where η and ζ are the un-normalized viscosity coefficients defined in section 2.2. The coefficients ν and μ will be used from now on, and they are called kinematic and bulk viscosity respectively.

The above partial differential equations can be written in expanded form as follows,

$$\frac{\partial\rho}{\partial t} + \frac{\partial(\rho V_x)}{\partial x} + \frac{\partial(\rho V_y)}{\partial y} + \frac{\partial(\rho V_z)}{\partial z} = 0 \tag{2.52}$$

$$\frac{\partial(\rho V_x)}{\partial t} + \frac{\partial(\rho V_x V_x)}{\partial x} + \frac{\partial(\rho V_x V_y)}{\partial y} + \frac{\partial(\rho V_x V_z)}{\partial z} + \frac{\partial(c_s^2\rho)}{\partial x} - \nu\rho\,\nabla^2 V_x - \mu\rho\frac{\partial(\nabla\cdot\vec{V})}{\partial x} \;=\; 0 \tag{2.53}$$

$$\frac{\partial(\rho V_y)}{\partial t} + \frac{\partial(\rho V_x V_y)}{\partial x} + \frac{\partial(\rho V_y V_y)}{\partial y} + \frac{\partial(\rho V_y V_z)}{\partial z} + \frac{\partial(c_s^2 \rho)}{\partial y} - \nu\rho\,\nabla^2 V_y - \mu\rho\,\frac{\partial(\nabla \cdot \vec{V})}{\partial y} = 0 \tag{2.54}$$

$$\frac{\partial(\rho V_z)}{\partial t} + \frac{\partial(\rho V_x V_z)}{\partial x} + \frac{\partial(\rho V_y V_z)}{\partial y} + \frac{\partial(\rho V_z V_z)}{\partial z} + \frac{\partial(c_s^2 \rho)}{\partial z} - \nu\rho\,\nabla^2 V_z - \mu\rho\,\frac{\partial(\nabla \cdot \vec{V})}{\partial z} = 0 \tag{2.55}$$

where ρ, V_x, V_y, V_z are the fluid density and the components of the fluid velocity in the x,y,z directions respectively. The expanded form of the Laplacian operator ∇^2 is as follows,

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2} \tag{2.56}$$

A simplified form of the Navier Stokes equations can be obtained by omitting the bulk viscosity term $\mu\rho\partial(\nabla \cdot \vec{V})/\partial x$ because it is very small in the case of subsonic flow (see section 2.4.2). Also, the continuity equation 2.52 can be subtracted from each one of the momentum equations 2.53–2.55, and the equations can be divided by the density ρ. The resulting equations have the following form,

$$\frac{\partial \rho}{\partial t} + \frac{\partial(\rho V_x)}{\partial x} + \frac{\partial(\rho V_y)}{\partial y} + \frac{\partial(\rho V_z)}{\partial z} = 0 \tag{2.57}$$

$$\frac{\partial V_x}{\partial t} + V_x\frac{\partial V_x}{\partial x} + V_y\frac{\partial V_x}{\partial y} + V_z\frac{\partial V_x}{\partial z} + \frac{c_s^2}{\rho}\frac{\partial \rho}{\rho \partial x} - \nu\nabla^2 V_x = 0 \tag{2.58}$$

$$\frac{\partial V_y}{\partial t} + V_x\frac{\partial V_y}{\partial x} + V_y\frac{\partial V_y}{\partial y} + V_z\frac{\partial V_y}{\partial z} + \frac{c_s^2}{\rho}\frac{\partial \rho}{\rho \partial y} - \nu\nabla^2 V_y = 0 \tag{2.59}$$

$$\frac{\partial V_z}{\partial t} + V_x\frac{\partial V_z}{\partial x} + V_y\frac{\partial V_y}{\partial y} + V_z\frac{\partial V_z}{\partial z} + \frac{c_s^2}{\rho}\frac{\partial \rho}{\rho \partial z} - \nu\nabla^2 V_z = 0 \tag{2.60}$$

The next section discusses the significance of the shear and the bulk viscosity terms.

2.4.1 Shear viscosity

The coefficient ν that appears in the Navier Stokes equations is called the kinematic viscosity. It is equal to the first coefficient of viscosity η divided by the mean density of the fluid medium. The coefficient η varies very slowly with temperature, and the coefficient ν varies very slowly both with temperature and with density. The

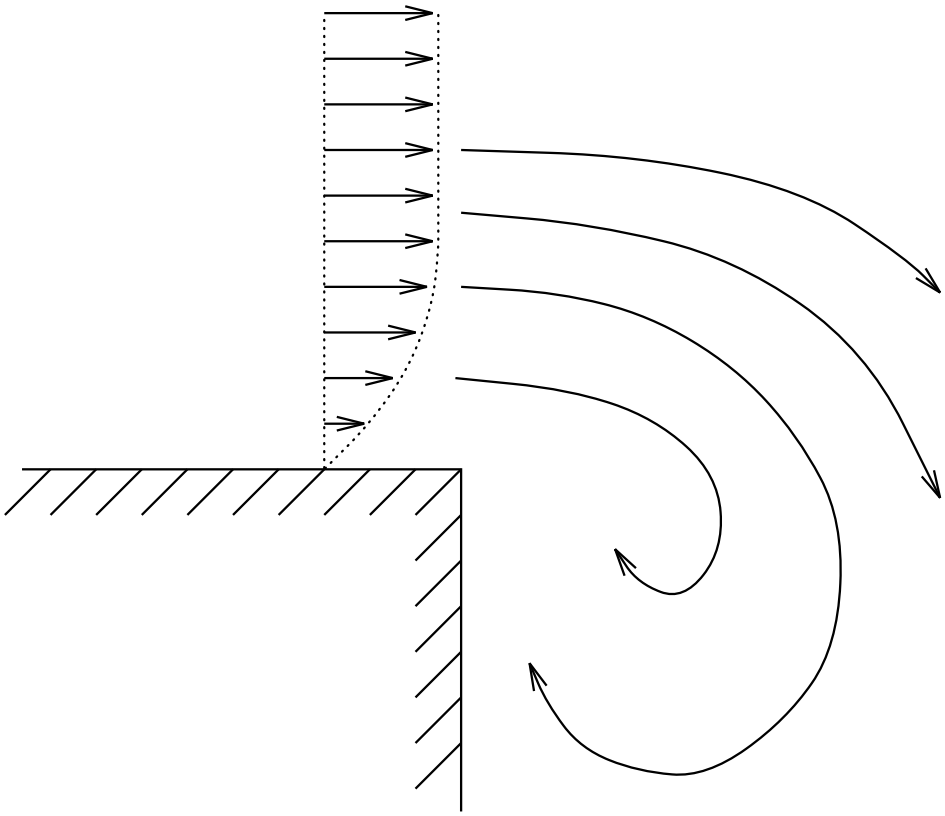

Figure 2-2: A vortex forms when the flow bends around a sharp corner. If the flow speed is large, the vortex may separate and move away with the flow, while new vortices are being formed in its place.

variations are so small, however, that they are ignored here. Thus, ν is assumed to be constant. The value of ν at selected temperatures is given in section 2.6.

Physically, the ν term corresponds to friction, and it expresses the loss of momentum due to shearing forces in the fluid. For example, when two layers of fluid slide over each other with opposing velocities, or when a layer of fluid is moving over a flat plate that is stationary with respect to the flow (figure 2-4), the ν term is responsible for decelerating the neighboring layers of the fluid that move with different speeds. Generally, the ν term is responsible for smoothing and diffusing differences in the velocity of the fluid.

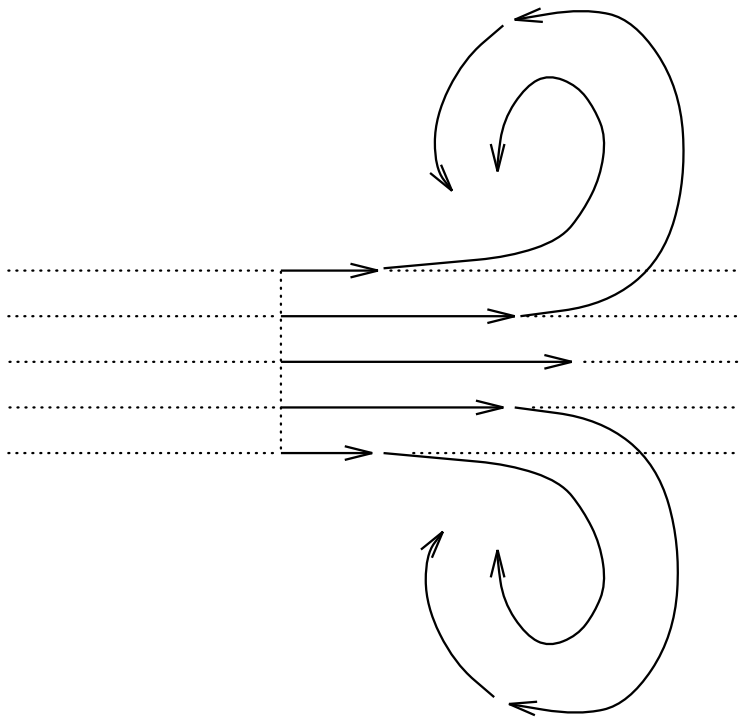

Figure 2-3: The boundary layer around a jet curls up and forms vortices that separate from the main jet.

An important property of viscous fluids is that when the ν coefficient is very small, the deceleration of the fluid due to viscosity occurs in small regions that are called boundary layers (Newman [34, pp.70–68]). Inside a boundary layer the flow velocity changes very rapidly from one value to another, which makes the velocity gradients very large, and thus the ν term of the Navier Stokes equations large enough that it *can not* be neglected. Figure 2-4 shows the boundary layer above a flat plate, where the plate is stationary with respect to a fast-moving flow. The speed of the fluid changes from zero at the surface of the flat plate to some large value away from the plate on the other side of the boundary layer.

In contrast to the above discussion of narrow boundary layers, I must clarify that very thick boundary layers are also possible, although in this case the name "boundary layer" is usually avoided. In particular, the boundary layers can grow slowly in space and in time by diffusion, so that they can become very large in steady flow if the solid boundary extends for a long distance, and if sufficient time elapses for the boundary layer to grow from an initial non-moving state. An example of this situation is the Hagen-Poiseuille flow (Newman [34, p.63–85], Landau&Lifshitz [32, p.51]) inside long pipes, where the velocity assumes a parabolic profile eventually, and the boundary layer can be considered to extend the radius of the pipe. However, in the case of unsteady flow, and in the case where the solid boundary has a limited extent (a finite obstacle), the boundary layer can not grow, and it remains a narrow boundary layer around the solid obstacle as described in the previous paragraph.

Under appropriate conditions, such as fast flow around sharp corners, the boundary layer separates from the region where it is formed, and begins "to take a life of its own" as it moves away with the flow. As soon as it separates, the boundary layer turns around itself and forms narrow loops of turning flow, which are called vortices. A simple way to understand this curling up is that the different sides of the boundary layer are moving with different speeds, so that when the two sides are suddenly free, they can only turn into themselves and curl up. Figure 2-2 shows the formation of a vortex around a sharp corner, and figure 2-3 shows the formation of vortices around a jet that is injected at high speed into a stationary fluid.

The angular speed of a vortex can be calculated using the curl of the fluid velocity, which is called the vorticity. In three dimensions the curl of the velocity is a 3D vector, while in two dimensions the curl of the velocity is simply a scalar with a direction normal to the plane of the flow, for example the z-axis. In particular, the following formula applies,

$$(\vec{\nabla} \times \vec{V})_z = \left(\frac{\partial V_y}{\partial x} - \frac{\partial V_x}{\partial y} \right) \tag{2.61}$$

In chapter 7, contour plots of the above scalar vorticity are used in order to visualize

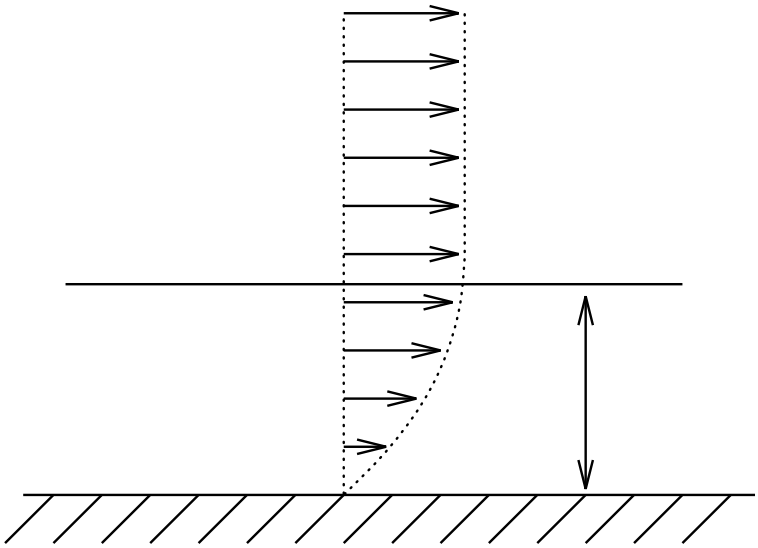

Figure 2-4: A boundary layer forms above a flat plate that is stationary with respect to a fast-moving flow.

the flow.

It is worth mentioning that a common approximation in fluid mechanics is to assume that the vorticity is zero for the most part of the flow. The rationale behind this approximation is to assume that the viscosity is very small so that it can neglected for the most part of the flow (inviscid flow). Also, the fluid is assumed to be initially at rest so that the vorticity is initially zero. Then, according to Kelvin's circulation theorem for inviscid flows, the integral of vorticity remains constant in time over any simply-connected surface of the flow (Newman [34, p.105]). Physically, this means (Tritton [54] p.84 and p.114) that if the vorticity is initially zero, it will always remain zero. Of course, the viscosity can not be neglected in boundary layers where the velocity gradient is large. Thus, the condition of zero vorticity is always an approximation which we hope is valid for the most part of the flow.

The condition of zero vorticity enables us to write the velocity field as the gradient of a scalar potential function. For example, in two dimensions, the condition of zero vorticity implies that

$$\frac{\partial V_y}{\partial x} = \frac{\partial V_x}{\partial y} \tag{2.62}$$

This is the condition of integrability for an exact differential (Courant [13, p.353]),

$$d\phi = V_x\, dx + V_y\, dy \tag{2.63}$$

Therefore, the scalar potential function ϕ can be introduced in place of the vector velocity. The scalar potential function is very useful because it enables us to calculate analytically the solutions of many flow geometries (see chapter 4 of Newman's book [34]) especially in two dimensions.

Because of its versatility in finding analytic solutions, the potential approximation is used very widely, even when there is a lot of vorticity in the flow, and the inviscid assumption is very questionable. The way this is done is by introducing a potential function with singularities where the vorticity is zero everywhere except at a few singular points called point vortices. Using such techniques, the effect of boundary layers, and also the effect of unsteady generation of vorticity can be handled within the framework of a potential model (chapter 4 of Newman's book [34]). The potential model is also useful in situations which are too complex to analyze otherwise, and the potential model provides at least one estimate of the behavior of the flow. Such an example is the flow near the edge of a flue pipe (for an overview see Verge94 [56] and Hirschberg94 [26]). The success of the potential model in these situations depends on having a good understanding of the flow in order to make the right assumptions and the right approximations.

The above discussion on potential flow, vortex theory, and boundary layers is not critical for the computer simulations, but it is useful background material. All of the ideas introduced above are very important parts of fluid dynamics, and there are entire books and chapters devoted to their study [34, 44, 3, 54].

2.4.2 Bulk viscosity

The μ term of the Navier Stokes equations 2.53–2.55 is called bulk viscosity, and expresses the loss of momentum during elastic compression and dilatation of fluid elements. The actual value of the μ coefficient is difficult to measure experimentally, and is not known for many types of fluids (Tritton [54, p.58]). It is common practice to use the following value,

$$\mu = \frac{1}{3}\nu \tag{2.64}$$

which is called the Stokes' relation and corresponds to setting the second coefficient of viscosity equal to zero $\zeta = 0$ in equation 2.51 (Peyret&Taylor [38, p.11] and Tritton [54, p.58]).

In the case of subsonic flow, the μ term is often omitted because the gradient of the divergence of velocity is very small compared to the other terms of the momentum Navier Stokes equations (see below). Accordingly, in the computer simulations I omit the μ term when I use finite difference methods. However, when I use the lattice Boltzmann method, I employ a positive μ term which comes with the lattice Boltzmann method by construction. The value of the μ coefficient for the lattice Boltzmann method (two-dimensional orthogonal model) is given by equation 4.47 of chapter 4, and depends on two lattice Boltzmann parameter w_0 and y_0. The parameter y_0 is usually chosen $y_0 = w_0/4$ and the resulting formula for μ is,

$$\mu \;=\; \nu\,(2 \;-\; 9\,w_0) \tag{2.65}$$

There is considerable freedom in choosing w_0 within the constraints $w_0 > 0$ and $5\,w_0 + z_0 = 1$ where $z_0 > 0$. For example, the value $w_0 = 5/27$ produces the Stokes relation $\mu = \frac{1}{3}\nu$ (see section 4.1.3 regarding the maximum value of w_0 for stability). In my simulations, I also use the values $w_0 = 1/7$ and $w_0 = 1/6$ which produce slightly larger values of μ than the Stokes relation. I do not pay much attention to the precise value of μ because the μ term is very small in subsonic flow.

The reason why the μ term is very small in subsonic flow compared to the other

terms of the momentum Navier Stokes equations is as follows. The continuity equation shows that the divergence of velocity is directly proportional to changes in density. The momentum equation shows that changes in density are proportional to changes in fluid momentum after multiplication by the square of the speed of sound. Since the speed of sound is much larger than the fluid speed in subsonic flow, the gradient of the divergence of velocity (the μ term) is expected to be small compared to the other terms in the momentum equation.

The above argument can be made precise by obtaining an estimate for the gradient of the divergence of velocity from the continuity equation 2.53, as follows,

$$-\frac{\partial(\nabla \cdot \rho\vec{V})}{\partial x} = \frac{\partial(\partial\rho/\partial t)}{\partial x} \tag{2.66}$$

The above estimate of the divergence of velocity (multiplied by μ) must be compared against the other terms of the momentum equation. Let us choose the pressure term as a good representative of the size of the terms in the momentum Navier Stokes equations. We inquire whether the following inequality is true,

$$\mu\frac{\partial}{\partial t}\left(\frac{\partial\rho}{\partial x}\right) \ll c_s^2\left(\frac{\partial\rho}{\partial x}\right) \tag{2.67}$$

where the symbol "$\ll$" means "very small compared to". To prove this inequality, we can estimate that the time derivative of $\partial\rho/\partial x$ can not be larger than the present value of $\partial\rho/\partial x$ times the speed of sound divided by some wavelength λ that corresponds to this disturbance. This is because the fastest changes in subsonic flow propagate at the speed of sound. Thus, we can write,

$$\mu\frac{\partial}{\partial t}\left(\frac{\partial\rho}{\partial x}\right) \leq \mu\left(\frac{c_s}{\lambda}\right)\left(\frac{\partial\rho}{\partial x}\right) \tag{2.68}$$

From the above, we can conclude that in the case of subsonic flow the inequality 2.67 is equivalent to the following inequality,

$$\mu\left(\frac{c_s}{\lambda}\right) \ll c_s^2 \tag{2.69}$$

or

$$\mu \ll c_s \lambda \tag{2.70}$$

In the case of air, the bulk viscosity coefficient μ of air is about 0.075 cm^2/s (using $\mu = \nu/2$), and the speed of sound is $c_s = 34400$ cm/s (see section 2.6). The smallest wavelength (smallest length of disturbance) at which the Navier Stokes equations are applicable is about $\lambda = 10^{-3}$ cm as explained in section 2.1, so the above inequality is well satisfied. Also, the wavelength $\lambda = 10^{-3}$ cm corresponds to a frequency $f = c_s/\lambda = 34$ MHz which is well beyond the range of acoustic frequencies that we are interested in the case of musical instruments, namely less than 20 kHz. Therefore, it is reasonable to ignore the μ term in the computer simulations of flue pipes. In section 2.5.2, the effect of bulk viscosity on the decay of acoustic waves is calculated exactly, and is found to be extremely small.

2.4.3 Incompressible flow approximation

This section describes the incompressible flow approximation of the Navier Stokes equations. This approximation is not used in the computer simulations, but is useful background material.

First, a word on terminology is in order. An incompressible flow is also called "hydrodynamic flow" in view of the fact that the compressibility of water is very small. Of course, this is only a naming convention, and does not imply that water is perfectly incompressible which is false. Further, the term "hydrodynamic" is also used to distinguish the dynamics of a flow which do *not* depend on compressible effects (the hydrodynamics) from the dynamics of a flow which do depend on compressible effects (the acoustic waves). In other words, the name "hydrodynamic" is general term, and is somewhat different from the precise notion of incompressibility which is the subject of this section.

In incompressible flow, the continuity equation is replaced with the condition that

the divergence of the velocity is zero,

$$\frac{\partial V_x}{\partial x} + \frac{\partial V_y}{\partial y} + \frac{\partial V_z}{\partial z} \;=\; 0 \tag{2.71}$$

Also, the term $(c_s^2/\rho)\,\partial\rho/\partial x$ is usually written as $\partial P/\partial x$ so that the density variable does not appear at all in the equations of fluid flow. Aside from this change, the momentum equations 2.58–2.60 remain unchanged in all other respects.

The rationale behind the incompressible flow idea is to assume that the time derivative and the spatial variations of the density are very small compared to the velocity gradients. Such an assumption originates from the fact that the density gradient $\partial\rho/\partial x$ is proportional to the derivatives of velocity divided by the square of the speed of sound, (see the momentum Navier Stokes equations). Because the ratio V/c_s is very small in the case of subsonic flow, the density gradient is very small compared to the derivatives of the velocity.

Physically, the condition of incompressibility (zero divergence of the velocity) implies that any disturbances of the fluid propagate with infinite speed to other parts of the fluid. This is only an approximation because any disturbances of a real compressible fluid propagate with a finite speed of sound to other parts of the fluid via acoustic waves. The advantage of assuming an infinite speed of sound propagation is to allow us to solve the Navier Stokes equations without having to follow the propagation of acoustic waves step by step. Thus, the numerical solutions of the Navier Stokes equations can be speeded up, and the theoretical analysis of the equations can be simplified in many cases as well.

Another way of understanding the incompressible flow approximation is to consider the following situation which appears paradoxical at first sight. A solution of the incompressible flow equations is steady flow inside long pipes, known as Hagen-Poiseuille flow (Landau&Lifshitz [32, p.51]). Let us consider a two-dimensional pipe with the walls located at $y = 1$ and $y = 0$, and let us assume a constant pressure

gradient along x,

$$\frac{\partial P}{\partial x} \;=\; \frac{c_s^2}{\rho_0}\,\frac{\partial \rho}{\partial x} \;=\; \text{constant} \tag{2.72}$$

where the adiabatic relation between pressure and density is used. As stated earlier, the density gradient is extremely small because the speed of sound is extremely large compared to the flow speed; which is the reason why we can neglect the density variations from the continuity equation and arrive at the incompressible flow condition $\nabla \cdot \vec{V} = 0$. The steady state solution of the above problem is a vanishing vertical velocity $v = 0$, and a parabolic profile for the horizontal velocity u,

$$u(y,t) \;=\; \frac{1}{2}\left(\frac{c_s^2}{\nu\,\rho_0}\,\frac{\partial \rho}{\partial x}\right) y\,(y-1) \tag{2.73}$$

By substitution, we can show that the above solution satisfies the momentum Navier Stokes equations, the condition of incompressibility, and the boundary conditions of vanishing velocity at the walls. However, a paradox arises when we try to substitute the above solution into the continuity equation 2.57. The continuity equation expresses the conservation of fluid, and it must be satisfied always independent of the approximations that we introduce. In expanded form we have,

$$\frac{\partial \rho}{\partial t} + u\,\frac{\partial \rho}{\partial x} + v\,\frac{\partial \rho}{\partial y} + \rho\,\frac{\partial u}{\partial x} + \rho\,\frac{\partial v}{\partial y} \;=\; 0 \tag{2.74}$$

All the terms except $u\,\partial\rho/\partial x$ vanish according to our solution, therefore the term $u\,\partial\rho/\partial x$ must vanish also. This is an apparent contradiction because we know that the density gradient $\partial\rho/\partial x$ must be very small, but not identically zero. The term $u\,\partial\rho/\partial x$ expresses the change of density which is caused by the flow, and from a physical point of view there must be another term that balances this change of density, no matter how small it may be. The question is "which term balances the change of density?"

One way of resolving the paradox is to add a correction to the $\rho\partial u/\partial x$ term in order to balance equation 2.74. This is a reasonable assumption in view of the steadiness and symmetry of the problem along the y direction. Thus, we assume

that the horizontal velocity has a very small but non-vanishing variation in x, and we introduce a modified velocity $\tilde{u}$ with a correction $\epsilon(x)$,

$$\tilde{u} \;=\; u\,[1 \,+\, \epsilon(x)] \tag{2.75}$$

The continuity equation is satisfied to first order in ϵ (that is, the error is of order ϵ^2) if the correction ϵ is as follows,

$$\epsilon(x) \;=\; -\left(\frac{1}{\rho_0}\,\frac{\partial \rho}{\partial x}\right)\,x \tag{2.76}$$

Thus, we have found that the horizontal velocity has a very small but non-vanishing variation in x. Our original solution must be modified according to the following formula,

$$u(y,t) \;=\; \frac{1}{2}\,\left(\frac{c_s^2}{\nu\,\rho_0}\,\frac{\partial \rho}{\partial x}\right)\,y\,(y-1)\,\left[1 \,-\, \left(\frac{1}{\rho_0}\,\frac{\partial \rho}{\partial x}\right)\,x\right] \tag{2.77}$$

The importance of the above correction is to helps us understand better the approximation of incompressible flow. In practice, the correction is not very useful because it is exceedingly small. In particular, if the size of the velocity is of order unity $u \sim 1$ cm/s, then the normalized density gradient $(1/\rho_0)\partial\rho/\partial x$ is of the order $1/c_s^2$ which is about 10^{-9} in air at room temperature and atmospheric pressure. If we can measure the flow velocity with an accuracy of 10^{-3} (3 decimals), then the above correction to the velocity becomes noticeable between the ends of a pipe that is 10 km long. This is of course unrealistic because other effects that we have not modeled here become important in a 10 km pipe.

The above analysis of Hagen-Poiseuille flow in a pipe is an example where the incompressible flow approximation works very well. By contrast, the flow of air inside a wind musical instrument is an example where the acoustic waves interact with the hydrodynamics of the flow. In such a situation, the incompressible flow approximation is inapplicable, and the compressible Navier Stokes equations must be used instead. Below, the wave equation is discussed because it is useful background material for the simulations of flue pipes.

2.5 The wave equation

In this section, the wave equation is derived from the compressible Navier Stokes equations, and further some interesting solutions of the wave equation are described. An acoustic wave in a compressible medium is usually defined as an oscillatory motion of small amplitude [32, p.251]. The oscillation arises through the interchange of energy between the kinetic and the potential forms; namely, the velocity and the density (pressure) of the compressible medium. By means of this oscillatory mechanism, any disturbance of the density and/or the velocity of the compressible medium propagates inside the medium and reflects off boundaries. The speed of propagation is characteristic of the medium, and it is called the speed of sound.

2.5.1 Linear inviscid

In order to derive the wave equation from the Navier Stokes equations, we consider the simplest situation from an acoustic point of view. Namely, we assume that the mean flow is zero, and that the amplitude of the acoustic disturbance is small. Mathematically, this means that the fluid velocity and density can be written as follows,

$$\begin{aligned} \rho &= \rho_0 + \rho' \qquad \rho_0 \text{ constant}, \quad \rho' \ll \rho_0 \\ \vec{V} &= \vec{V}_0 + \vec{V}' \qquad \vec{V}_0 = 0, \qquad \vec{V}' \text{ small} \end{aligned} \tag{2.78}$$

We also consider the compressible Navier Stokes equations in their original form,

$$\frac{\partial \rho}{\partial t} + (\vec{V} \cdot \vec{\nabla}\rho) + \rho(\nabla \cdot \vec{V}) = 0 \tag{2.79}$$

$$\frac{\partial V_j}{\partial t} + V_k \frac{\partial V_j}{\partial x_k} + \frac{c_s^2}{\rho}\frac{\partial \rho}{\partial x_j} - \nu \nabla^2 V_j - \mu \frac{\partial(\nabla \cdot \vec{V})}{\partial x_j} = 0 \tag{2.80}$$

where $j = 1, 2, 3$ stands for the Cartesian directions x, y, z. If we substitute the density and the velocity given by equation 2.78 in the above Navier Stokes equations, and neglect small terms that are quadratic in the acoustic amplitude, we obtain,

$$\frac{\partial \rho'}{\partial t} + \rho_0 \nabla \cdot \vec{V}' = 0 \tag{2.81}$$

$$\frac{\partial V_j'}{\partial t} + \frac{c_s^2}{\rho_0}\frac{\partial \rho'}{\partial x_j} = 0 \tag{2.82}$$

In the above calculation, the quadratic terms $(\vec{V}' \cdot \vec{\nabla}\rho')$ and $V_k' \partial V_j'/\partial x_k$ are omitted from the continuity and the momentum equations respectively. Also, the approximation $(\rho_0 + \rho') \sim \rho_0$ is used whenever the density ρ appears as a multiplicative factor. Further, the viscosity terms are omitted because they have a negligible effect on the acoustic waves, as we shall see in section 2.5.2. These may seem a lot of simplifications, but they are very reasonable for sound waves of small amplitude in air.

To proceed further, we try to obtain an equation involving the density only. We differentiate the continuity equation 2.81 with respect to time, and the momentum equation 2.82 with respect to the spatial direction x_j in order to eliminate the velocity from the density equation. In two dimensions we have three equations. We use the notation $u = V_1$ and $v = V_2$ for the x, y components of the velocity,

$$\frac{\partial^2 \rho'}{\partial t^2} + \rho_0 \left(\frac{\partial^2 u}{\partial x \partial t} + \frac{\partial^2 v}{\partial y \partial t} \right) = 0 \tag{2.83}$$

$$\frac{\partial^2 u}{\partial x \partial t} + \frac{c_s^2}{\rho_0}\frac{\partial^2 \rho'}{\partial x^2} = 0 \tag{2.84}$$

$$\frac{\partial^2 v}{\partial y \partial t} + \frac{c_s^2}{\rho_0}\frac{\partial^2 \rho'}{\partial y^2} = 0 \tag{2.85}$$

By subtracting the above momentum equations from the continuity equation, we obtain a linear wave equation for the acoustic density,

$$\frac{\partial^2 \rho'}{\partial t^2} - c_s^2 \left(\frac{\partial^2 \rho'}{\partial x^2} + \frac{\partial^2 \rho'}{\partial y^2} \right) = 0 \tag{2.86}$$

A complementary approach is to try to obtain an equation involving the acoustic velocity only. To do so, we differentiate the continuity equation 2.81 with respect to x and y, and the momentum equation 2.82 with respect to time. Then, we can eliminate the density from the velocity equations to obtain,

$$\frac{\partial^2 u}{\partial t^2} - c_s^2 \frac{(\nabla \cdot \vec{V}')}{\partial x} = 0 \tag{2.87}$$

$$\frac{\partial^2 v}{\partial t^2} - c_s^2 \frac{(\nabla \cdot \vec{V}')}{\partial y} = 0 \tag{2.88}$$

or equivalently,

$$\frac{\partial^2 u}{\partial t^2} - c_s^2 \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 v}{\partial x \partial y} \right) = 0 \tag{2.89}$$

$$\frac{\partial^2 v}{\partial t^2} - c_s^2 \left(\frac{\partial^2 u}{\partial x \partial y} + \frac{\partial^2 v}{\partial y^2} \right) = 0 \tag{2.90}$$

The above appear to be coupled equations in x, y; however, the fact that we have omitted viscosity from our acoustic model (see equation 2.82) can actually decouple the above equations. By calculating the curl of equation 2.82, we can show that the vorticity of the acoustic flow is constant, as follows,

$$\frac{\partial}{\partial t} \left(\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} \right) \;=\; \frac{c_s^2}{\rho_0} \left(\frac{\partial^2 \rho'}{\partial x \partial y} - \frac{\partial^2 \rho'}{\partial y \partial x} \right) \;=\; 0 \tag{2.91}$$

Furthermore, a reasonable assumption for acoustic waves is that the acoustic motion starts from an initial state of rest, so that the vorticity is initially zero. The above equation shows that the vorticity remains always zero. Thus, the following condition of integrability is satisfied (Courant [13, p.353]),

$$d\phi \;=\; u\,dx \;+\; v\,dy \tag{2.92}$$

and the acoustic velocity is the gradient of a scalar potential function, denoted here by ϕ. The above acoustic potential is a special case of the general potential model (zero vorticity approximation) that is discussed in section 2.4.1. We have the relations,

$$\begin{aligned} \partial\phi/\partial x &= u \\ \partial\phi/\partial y &= v \\ \partial^2 u/(\partial x \partial y) &= \partial^3\phi/(\partial x^2 \partial y) = \partial^2 v/\partial y^2 \\ \partial^2 v/(\partial x \partial y) &= \partial^3\phi/(\partial x \partial y^2 = \partial^2 u/\partial y^2 \end{aligned} \tag{2.93}$$

By substituting the above into equations 2.89, 2.90, we obtain the linear wave equation for each component of the velocity,

$$\frac{\partial^2 u}{\partial t^2} - c_s^2 \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) = 0 \tag{2.94}$$

$$\frac{\partial^2 v}{\partial t^2} - c_s^2 \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right) = 0 \tag{2.95}$$

By substituting the potential ϕ in equation 2.82, and integrating along x_j as follows,

$$\int \left[\frac{\partial^2 \phi}{\partial t \partial x_j} + \frac{c_s^2}{\rho_0} \frac{\partial \rho'}{\partial x_j} \right] \, dx_j = 0 \tag{2.96}$$

we obtain

$$\frac{\partial \phi}{\partial t} + C(t) = - \left(\frac{c_s^2}{\rho_0} \right) \, \rho' \tag{2.97}$$

where $C(t)$ is an integration constant that can only depend on time. Therefore, it can be absorbed in the velocity potential without affecting the spatial gradient. For example, we can redefine the potential as follows (for a related problem see Newman [34, p.108]),

$$\phi' = \phi + \int^t C(\tau) \, d\tau \tag{2.98}$$

but we keep the same symbol ϕ below for simplicity. Thus, we obtain a simple relation between the acoustic density and the acoustic potential,

$$\rho' \; = \; - \left(\frac{\rho_0}{c_s^2} \right) \, \frac{\partial \phi}{\partial t} \tag{2.99}$$

and also a linear wave equation for the potential ϕ

$$\frac{\partial^2 \phi}{\partial t^2} - c_s^2 \left(\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} \right) = 0 \tag{2.100}$$

A typical solution of the above linear wave equation is a plane wave traveling along the positive x direction,

$$\begin{aligned} \phi(t,x,y) &= f(x - c\,t) \\ u(x,y,t) &= \overset{\circ}{f}\,(x - c\,t) \\ \rho'(t,x,y) &= (\rho_0/c) \, \overset{\circ}{f}\,(x - c\,t) \end{aligned} \tag{2.101}$$

where $f(x)$ is an arbitrary differentiable function of one variable, and $\overset{\circ}{f}\,(x)$ denotes its first derivative. The negative-traveling wave $f(x + c\,t)$ is also a solution. Because the wave equation is linear, any superposition of solutions is also a solution. In

particular, the complex exponentials that satisfy the wave equation can be used to represent almost any solution as a sum (integral) of complex exponentials according to Fourier's decomposition theorem (Courant [13, p.318]). The complex exponentials that satisfy the wave equation are as follows,

$$u(x,y,t) \;=\; e^{i(k\,x\,-\omega\,t)} \tag{2.102}$$

The above complex exponential is a *periodic traveling wave* that advances in the positive x direction with increasing time. The speed of propagation is the speed of sound c_s, and we have the following relations,

$$c_s \;=\; \frac{\omega}{k} \;=\; f\,\lambda \tag{2.103}$$

$$\lambda \;=\; \frac{2\,\pi}{k} \tag{2.104}$$

$$\omega \;=\; 2\,\pi\,f \tag{2.105}$$

where k is the spatial frequency, λ is the wavelength, f is the time frequency in cycles per second (Hz), and ω is the time frequency in radians per second. Because of the linearity of the wave equation, it is valid to calculate with complex exponentials which are more convenient than sines and cosines, as long as we are consistent in taking the real part (or the imaginary part) of our expressions at the beginning and the end of the calculation. For example, the complex exponential solution of equation 2.102 "contains" the following two physical solutions,

$$\begin{aligned} u(x,y,t) &= \cos\left(k\,x\;-\omega\,t\right) \\ u(x,y,t) &= \sin\left(k\,x\;-\omega\,t\right) \end{aligned} \tag{2.106}$$

depending on whether we choose the real or the imaginary part.

Apart from traveling waves, there are also *stationary waves* which means that the time variation is decoupled from the spatial variation. Stationary waves arise when we impose fixed boundary conditions such as two walls at which the acoustic velocity must vanish. The simplest way to construct a *stationary wave* is to combine two

periodic traveling waves that are identical except for traveling in opposite directions. For example,

$$\frac{1}{2}\left(e^{i(k\,x+\omega\,t)}+e^{i(k\,x-\omega\,t)}\right) = e^{i(k\,x)}\cos(\omega\,t) \tag{2.107}$$

which corresponds to the following two stationary waves,

$$\begin{aligned} u(x,t) &= \cos(k\,x)\cos(\omega\,t) \\ u(x,t) &= \sin(k\,x)\cos(\omega\,t) \end{aligned} \tag{2.108}$$

A stationary wave possesses *nodes and loops* that are fixed in space. A node is a point where the amplitude vanishes, while a loop is a point where the amplitude achieves maximum values during one period of oscillation. In the case of stationary waves, the velocity nodes are density loops, and the density nodes are velocity loops. To see this, we calculate the density that corresponds to the velocity of equation 2.108 by differentiating in space and integrating in time as prescribed by the continuity equation 2.81. We obtain,

$$\begin{aligned} \rho'(x,t) &= -\rho_0\,(k/\omega)\sin(k\,x)\sin(\omega\,t) \\ \rho'(x,t) &= \rho_0\,(k/\omega)\cos(k\,x)\sin(\omega\,t) \end{aligned} \tag{2.109}$$

We see that the loops and nodes are interchanged between density and velocity in the case of stationary waves. This is in contrast to free sinusoidal traveling wave (equation 2.101) where the loops and nodes occur at the same locations for the velocity and the density, and furthermore the loops and nodes are moving with time.

Finally, it should be noted that the solutions of the wave equation 2.82 for a compressible medium such as air are longitudinal waves in the sense that the acoustic velocity oscillates along the same direction as the direction of wave propagation. By contrast, the sound waves of a violin string are transversal oscillations in the sense that the acoustic motion is at right angles to the direction of propagation along the string. We can examine mathematically the longitudinal character of the wave equation 2.82 by trying to find a transversal solution as follows,

$$\begin{aligned} u(x,y,t) &= u(y,t) \\ v(x,y,t) &= v(x,t) \end{aligned} \tag{2.110}$$

Then, the divergence of velocity is identically zero. The x momentum equation 2.82 implies that the density must be constant with x because the velocity $u(y,t)$ only varies with y. Similarly, the y momentum equation 2.82 implies that the density must be constant with y because the velocity $v(x,t)$ only varies with x. The integration of the continuity equation 2.81 says that the density ρ' must be constant in time. Consequently, there can be no wave motion at all. In fact, if we integrate equation 2.82, we obtain

$$\begin{aligned} u(y,t) &= -c_s^2(\partial\rho'/\partial x)\,t + u(y,0) \\ v(x,t) &= -c_s^2(\partial\rho'/\partial y)\,t + v(x,0) \end{aligned} \tag{2.111}$$

which says that the velocity becomes infinite with time. In other words, there are no physically relevant solutions in this case. We note that if we include viscosity in the wave equation, then we can obtain transverse oscillations of velocity that are physically relevant. We will examine these in the next section. However, the density of these transverse waves is also constant in time, so that the transverse waves can not be considered to be sound waves.

Having introduced the linear wave equation and some of its solutions, we discuss in the next section the solutions of a modified wave equation that includes the effects of viscosity. The modified equation of the next section is still linear, and thus it is straightforward to solve analytically.

2.5.2 Viscous decay of sound

In this section, the effect of viscosity on sound waves is calculated exactly in the special case of one-dimensional plane waves. [1] To this end, we retain the viscous terms of the Navier Stokes equations that we omitted earlier in equation 2.82. In

[1] This problem is also discussed in a slightly different way in Rayleigh [42, p.317], Lamb [31, p.647], and Morse&Ingard [33, p.285].

place of equations 2.81, 2.82 we have the following,

$$\frac{\partial \rho'}{\partial t} + \rho_0 \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right) = 0 \tag{2.112}$$

$$\frac{\partial u}{\partial t} + \frac{c_s^2}{\rho_0} \frac{\partial \rho'}{\partial x} - \nu \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) - \mu \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 v}{\partial y \partial x} \right) = 0 \tag{2.113}$$

$$\frac{\partial v}{\partial t} + \frac{c_s^2}{\rho_0} \frac{\partial \rho'}{\partial y} - \nu \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right) - \mu \left(\frac{\partial^2 u}{\partial x \partial y} + \frac{\partial^2 v}{\partial y^2} \right) = 0 \tag{2.114}$$

In the above equations we retain both the shear and the bulk viscosity terms because they have comparable size in the case of acoustic waves. We also inquire whether we should retain the nonlinear advection terms $(u\,\partial u/\partial x)$; however, these terms are smaller than the viscous terms as we shall see in section 2.5.4, when the amplitude of the sound wave is sufficiently small, which we assume to be so.

We look for a plane wave solution with $v = 0$ and $u = u(x,t)$, so that the equations simplify as follows

$$\frac{\partial \rho'}{\partial t} + \rho_0 \left(\frac{\partial u}{\partial x} \right) = 0 \tag{2.115}$$

$$\frac{\partial u}{\partial t} + \frac{c_s^2}{\rho_0} \frac{\partial \rho'}{\partial x} - (\nu + \mu) \left(\frac{\partial^2 u}{\partial x^2} \right) = 0 \tag{2.116}$$

By differentiating and combining the above equations, we obtain

$$\frac{\partial^2 u}{\partial t^2} - \left(c_s^2 + \tilde{\nu} \frac{\partial}{\partial t} \right) \frac{\partial^2 u}{\partial x^2} = 0 \tag{2.117}$$

where we denote $\tilde{\nu} = (\nu + \mu)$ for brevity. We look for general solutions of the form,

$$u = e^{i\,(k\,x\, -\, \omega\, t)\, -\, \alpha\, t} \tag{2.118}$$

where k, ω, α are real numbers, and they correspond to a spatial frequency, a time frequency, and a time constant of exponential decay. By substituting the above into the wave equation 2.117, we obtain,

$$(i\,\omega + \alpha)^2 = -k^2 \left(c_s^2 - \tilde{\nu}\,(i\,\omega + \alpha) \right) \tag{2.119}$$

which can solved exactly to give,

$$\alpha \;=\; \frac{k^2\,\tilde{\nu}}{2} \tag{2.120}$$

$$\frac{\omega}{k} \;=\; c_s\sqrt{1\;+\;\frac{k^2\,\tilde{\nu}^2}{4c_s^2}} \tag{2.121}$$

Therefore, we have obtained a periodic traveling wave solution that decays with a time constant α given by equation 2.120. The density function that corresponds to the velocity of equation 2.118 can be calculated using the continuity equation, and it is as follows,

$$\rho' \;=\; \frac{i\,\alpha\;+\;\omega}{\alpha^2\;+\;\omega^2}\,(k\,\rho_0)\,e^{i\,(k\,x\;-\;\omega\,t)\;-\;\alpha\,t} \tag{2.122}$$

By superposing two periodic traveling solutions that travel in opposite directions (the opposite traveling solution can be obtained by negating the time frequency ω in the above equations), we obtain a stationary solution,

$$u \;=\; (1/2)\,e^{-\alpha t}\,\left(e^{i(kx\,-\,\omega t)}\;+\;e^{i(kx\,+\,\omega t)}\right) \;=\; e^{-\alpha t}\,e^{i\,k\,x}\,\cos(\omega\,t) \tag{2.123}$$

$$\rho' \;=\; \frac{k\,\rho_0\,e^{-\alpha\,t}}{\alpha^2\;+\;\omega^2}\,e^{i\,k\,x}\;(i\,\alpha\,\cos(\omega\,t)\;-\;i\,\omega\,\sin(\omega\,t)) \tag{2.124}$$

By taking the imaginary part of the above expressions, we can obtain the following stationary solution in real numbers,

$$u \;=\; e^{-\alpha t}\,\sin k\,x\,\cos(\omega\,t) \tag{2.125}$$

$$\rho' \;=\; \frac{k\,\rho_0\,e^{-\alpha\,t}}{\alpha^2\;+\;\omega^2}\;(-\omega\,\cos(k\,x)\,\sin(\omega\,t)\;+\;\alpha\,\cos(k\,x)\,\sin(\omega\,t)) \tag{2.126}$$

Returning now to equation 2.120 for the decay constant, we can see that large spatial frequencies (short wavelength) decay faster than small spatial frequencies (long wavelength). However, the decay is extremely slow even for very large spatial frequencies. To see this, we use the values $c_s = 34400$ cm/s and $\tilde{\nu} = (\nu + .5\nu) = 0.225$ cm^2/s, and we write

$$\frac{\omega}{k} \;=\; c_s\,\sqrt{1\;+\;\epsilon} \tag{2.127}$$

The correction ϵ is very small for all frequencies of interest,

$$\epsilon \;=\; \frac{k^2\,\tilde{\nu}^2}{4c_s^2} \;=\; k^2 \times 1.069 \times 10^{-11} \tag{2.128}$$

Therefore,

$$\frac{\omega}{k} \;\simeq\; c_s \tag{2.129}$$

In particular, ϵ is equal to $1/100$ when k is about $k = 3 \times 10^4$. The correction ϵ decreases quadratically with smaller frequencies, so we can safely approximate $\omega = c_s\,k$ for all spatial frequencies up to $k = 3{\times}10^4$. Furthermore, the frequency $k = 3{\times}10^4$ is larger than the maximum frequency $k = 6.28 \times 10^3$ at which the Navier Stokes equations are applicable (see section 2.1). Therefore, the relation $\omega = c_s\,k$ is valid for all frequencies of interest.

We can calculate how many cycles, denoted by N, it takes for a sinusoidal acoustic wave to decay to one-tenth of its original value by setting,

$$e^{-\alpha N 2\pi/\omega} \;=\; 1/10 \tag{2.130}$$

By combining the above relation with equation 2.120, and the relation $k = \omega/c_s$, we obtain a relation between the frequency of the acoustic wave and the number of cycles N it takes for the wave to decay to one-tenth of its initial value.

$$f = \frac{\omega}{2\pi} \;=\; \frac{1}{2\pi}\left(\frac{\ln 10}{2\pi N}\right)\frac{2c_s^2}{\tilde{\nu}} \;=\; \frac{6.135 \times 10^8}{N} \tag{2.131}$$

For example, a frequency of 1 kHz takes 613500 cycles to decay to one-tenth of its value. The duration of this decay is about 613.5 s and corresponds to about 210 km for a traveling wave. Viscous effects are more pronounced at higher frequencies, as we can see from the following table (we are considering air under conditions of standard temperature and pressure).

Very high frequencies (above 1 MHz) decay very quickly in contrast to frequencies in the low range less than 20 kHz which decay extremely slowly.

f	λ	N	time	distance
1 kHz	34 cm	613500	613.5 s	210 km
100 kHz	0.34 cm	6135	0.61 s	21 m
1 MHz	0.034 cm	613	0.006 s	21 cm
10 MHz	0.0034 cm	61	0.00006 s	0.21 cm

Table 2.1: Viscous decay of acoustic waves in free space.

It should be noted that an alternative way of obtaining the above result is to look for a solution which decays with distance instead of time. To find a solution that decays with distance, we expect that the decaying exponential in time,

$$e^{-\alpha t} \tag{2.132}$$

should be replaced by a decaying exponential in space,

$$e^{-\frac{\alpha}{c_s} x} \tag{2.133}$$

based on the relation $k\,x = \omega\,t$ for the propagation of a traveling wave. In fact, if we substitute a trial solution of the form (a decaying wave that travels to the right $x > 0$),

$$u \;=\; e^{i\,(k\,x \,-\, \omega\,t) \,-\, \beta\,x} \tag{2.134}$$

into the linear dissipative wave equation 2.117, we can show that β is equal to α/c_s as expected,

$$\beta \;=\; \frac{k\,\omega\,\tilde{\nu}}{2\,c_s^2} \;=\; \frac{k^2\,\tilde{\nu}}{2\,c_s} \;=\; \frac{\alpha}{c_s} \tag{2.135}$$

Although the decay in space is similar to the decay in time, the two solutions differ in some ways. Whereas the time solution corresponds to a free traveling wave or a standing wave, the space solution corresponds to a wave emanating from an oscillating boundary condition, and it expresses the viscous decay of sound with distance from the source. The algebra of the two solutions is different also. For the solution in space we have the equation,

$$-\,\omega^2 = c_s^2\,(\beta^2 \,-\, k^2) \,-\, 2\,i\,\beta\,k\,c_s^2 \,+\, i\,\tilde{\nu}\,\omega\,(k^2 \,-\, \beta^2) \,-\, 2\,\tilde{\nu}\,\omega\,\beta\,k \tag{2.136}$$

Equating the imaginary parts, we obtain a quadratic equation for β, and we choose the decaying solution for positive $x > 0$ versus an unphysically-growing solution,

$$\beta \;=\; \frac{-k\,c_s^2}{\tilde{\nu}\,\omega}\left[1 \;-\; \sqrt{1 \;+\; (\tilde{\nu}\,\omega/c_s^2)^2}\right] \;\simeq\; \frac{k\,\omega\,\tilde{\nu}}{2\,c_s^2} \;=\; \frac{\alpha}{c_s} \tag{2.137}$$

Equating the real parts, and using the above approximate β, we find ω in terms of k,

$$\frac{\omega}{k} \;=\; c_s\sqrt{1 \;+\; 3\,\frac{\tilde{\nu}^2\omega^2}{4\,c_s^4}} \;\simeq\; c_s \tag{2.138}$$

Thus, we have a solution valid for $x > 0$,

$$u \;=\; e^{i\,(k\,x \;-\; \omega\,t) \;-\; \beta\,x} \tag{2.139}$$

$$\rho' \;=\; \frac{k \;+\; i\,\beta}{\omega}\,\rho_0\,e^{i\,(k\,x \;-\; \omega\,t) \;-\; \beta\,x} \tag{2.140}$$

The above solution decays very slowly with distance x for frequencies less than 100 kHz, as discussed previously.

In the case of musical instruments (acoustic frequencies less than 20 kHz), the above decaying solutions play a very small role. In particular, there are other effects such as the expansion of a wave in space ($1/r^2$ in three-dimensional space) which can reduce the power of a traveling wave much sooner than the viscous decay considered above. In the case of waves enclosed within pipes, the dominant mechanisms of loss of acoustic energy are the exchange of heat with the walls, and also the transverse viscous forces as opposed to the longitudinal viscous forces that we have considered above. We will consider a simple example of transverse viscous forces below. The effect of heat transfer with the walls is ignored by the adiabatic model of sound which assumes no heat transfer. Thermal effects are discussed in Kittel&Kroemer [28, p.434]. Transverse friction in combination with thermal effects are discussed in Landau&Lifshitz [32, p.301], and also Morse&Ingard [33, p.286].

2.5.3 Shear waves

This section analyzes shear waves which are transverse waves as opposed to the longitudinal waves of the previous section. These shear waves are another solution of

the dissipative wave equations 2.112–2.114. They are not proper acoustic waves however, because they do not involve any oscillations in density. In addition, we will see that they are solutions of the incompressible Navier Stokes equations as well, without any assumptions of linear acoustics such as small velocity amplitude. Physically, the shear waves correspond to the flow that arises when a rigid plate performs tangential oscillations along its own plane, and the fluid above the plate follows the oscillations because of shear viscous forces. To obtain the shear waves mathematically, we look for solutions of the form

$$u(y,t) \qquad \text{and} \qquad v = 0 \tag{2.141}$$

If we substitute the above expressions in the dissipative wave equations 2.112–2.114, we obtain

$$\frac{\partial \rho'}{\partial t} = 0 \tag{2.142}$$

$$\frac{\partial u}{\partial t} = \nu \frac{\partial^2 u}{\partial y^2} - \frac{c_s^2}{\rho_0} \frac{\partial \rho'}{\partial x} \tag{2.143}$$

$$\frac{\partial \rho'}{\partial y} = 0 \tag{2.144}$$

Immediately, we conclude that ρ' does not vary with y and t. Further, since we are looking for a velocity $u(y,t)$ that does not vary with x, equation 2.143 implies that $\partial \rho'/\partial x$ is a constant. To be careful, we should actually consider the possibility that there are velocity variations in x which are extremely small but non-vanishing (see section 2.4.3 on the paradox of incompressible flow). Then, according to equation 2.143 the variations of the density gradient $\partial \rho'/\partial x$ must be even smaller than the velocity variations by a factor of $1/c_s^2$. Thus, we can safely conclude that the density gradient $\partial \rho'/\partial x$ is constant based on equation 2.143.

An alternative way of deriving equation 2.143 is to consider the incompressible Navier Stokes equations discussed in section 2.4.3 instead of the acoustic equations 2.112–2.114. If we assume a velocity of the form $u(y,t)$ and $v = 0$, the divergence of the velocity becomes zero, so that incompressibility is satisfied. Then, a substitution in the momentum Navier Stokes equations 2.58–2.60 produces the same

equation 2.143 that we obtained above in the acoustic approximation except that now we do not require linear acoustics that the velocity amplitude is small.

To proceed further and to solve equation 2.143, we observe that the equation is linear so that different solutions can be superimposed. One solution follows by letting the time derivative of the velocity be zero; that is, by assuming steady flow. Then, we obtain the Hagen-Poiseuille flow that we discussed earlier in section 2.4.3, and has the form

$$u(y,t) \;=\; \frac{1}{2\,A}\left(\frac{c_s^2}{\nu\,\rho_0}\,\frac{\partial \rho}{\partial x}\right) A\,y^2 \;+\; B\,y \tag{2.145}$$

for arbitrary constants A and B that can be used to satisfy boundary conditions. This solution can be superimposed with the shear wave solution that we obtain immediately below.

The shear wave solution can be obtained by setting the density gradient, which is constant as we argued above, equal to zero, and by substituting a trial solution of the form

$$u(y,t) \;=\; e^{i\,(k\,y \;-\; \omega\,t)} \;-\; \beta\,y \tag{2.146}$$

We find,

$$-\,i\,\omega \;=\; \nu\,(\beta^2 \;-\; k^2 \;-\; i\,2\,\beta\,k) \tag{2.147}$$

which can be solved exactly to give,

$$\beta \;=\; \frac{\omega}{2\,k\,nu} \quad \text{and} \quad \beta \;=\; k \;=\; \sqrt{\frac{\omega}{2\,\nu}} \tag{2.148}$$

$$u(y,t) \;=\; \left(e^{-\sqrt{\frac{\omega}{2\,\nu}}\,y}\right)\left(e^{i\sqrt{\frac{\omega}{2\,\nu}}\,y}\right)\left(e^{-i\,\omega\,t}\right) \tag{2.149}$$

If we impose the boundary condition that there is a solid wall at $y = 0$ which is oscillating at a frequency of ω radians per second uniformly along its own plane (the x-axis), then the above expression becomes a simple shear wave,

$$u(y,t) \;=\; e^{-\sqrt{\frac{\omega}{2\,\nu}}\,y}\;\cos(\sqrt{\frac{\omega}{2\,\nu}}\,y \;-\; \omega\,t) \tag{2.150}$$

The above shear wave decays very fast with increasing distance from the oscillating boundary. In particular, the penetration depth at which the amplitude decreases by a factor of 10 is given by,

$$\delta \;=\; (\ln 10)\,\sqrt{\frac{2\,\nu}{\omega}} \tag{2.151}$$

For a frequency of 1 kHz in air at room temperature and atmospheric pressure the penetration depth is about 1.6×10^{-2} cm which is a very small distance.

The above shear waves provide an estimate of the viscous boundary layer of acoustic waves that are traveling along the length of a pipe. An plane wave inside a horizontal pipe oscillates back and forth along the x direction and creates friction against the walls in an analogous way to the oscillating shear waves above. Of course, there is one difference that the acoustic waves oscillate sinusoidally in x as opposed to uniformly in x that we considered above for an oscillating wall. Nevertheless, the penetration depth that we calculated above provides an approximate estimate of the viscous boundary layer of acoustic waves. It also shows that the effects of shear friction are much more pronounced than the effects of longitudinal friction that we considered in the previous section because the shear wave decays to zero within a very short distance in contrast to the longitudinal decay.

Another application of the shear wave solution that we obtained above is the testing of numerical methods, as we will see in section 4.5. In particular, for the purpose of numerical testing it is convenient to impose two boundary conditions: a non-moving wall at $y = 0$, and an oscillating plate at $y = 1$. We can satisfy these boundary conditions (Landau&Lifshitz [32, p.45]) if we constrain the general shear wave solution given by equation 2.149 to be of the form,

$$u(y,t) \;=\; e^{-\,i\,\omega\,t}\,\frac{\sin\left[(1+i)\,\sqrt{\dfrac{\omega}{2\,\nu}}\,y\right]}{\sin\left[(1+i)\,\sqrt{\dfrac{\omega}{2\,\nu}}\right]} \tag{2.152}$$

By expanding the sines of imaginary quantities in terms of hyperbolic sines and cosines, and performing some algebra, we can obtain a real solution for the problem

of an oscillating plate above a non-moving plate. The solution is presented and is used for numerical testing purposes in section 4.5.

In the next section, the relative size of the acoustic terms is examined when an acoustic wave is substituted into the Navier Stokes equations. This will confirm some of the discussions in the previous sections, for example the small effects of viscosity on acoustic waves, and it will also point to the limitations of the linear acoustic theory.

2.5.4 Relative size of acoustic terms

In order to estimate the relative size of the acoustic terms in a complete wave equation that includes both nonlinear advective terms and viscous terms, we consider a one-dimensional Navier Stokes equation for the velocity (Morse&Ingard [33, p.862]),

$$\frac{\partial u}{\partial t} + u\,\frac{\partial u}{\partial x} + \frac{c^2}{\rho_0}\,\frac{\partial \rho}{\partial x} + \tilde{\nu}\,\frac{\partial^2 u}{\partial x^2} \;=\; 0 \tag{2.153}$$

We substitute a typical sinusoidal wave in the above equation where u_0 is the wave amplitude,

$$\begin{aligned} u(x,t) &= \quad u_0\, e^{i(kx-\omega t)} \\ \rho'(x,t) &= (\rho_0/c)\, u_0\, e^{i(kx-\omega t)} \end{aligned} \tag{2.154}$$

We can estimate the relative size of the terms, as follows, where we normalize against the size of the first term,

$$\begin{array}{cccc} (\partial u/\partial t) & (u\,\partial u/\partial x) & ((\rho_0/c)\,\partial \rho'/\partial x) & (\tilde{\nu}\,\partial^2 u/\partial x^2) \\ u_0\,\omega & u_0^2\,k & u_0\,k\,c & u_0\,k^2\,\tilde{\nu} \\ 1 & u_0/c & 1 & k^2\,\tilde{\nu}/c \end{array} \tag{2.155}$$

If we compare the nonlinear advective term and the viscous term above, we obtain the result,

$$\frac{u\,\partial u/\partial x}{\tilde{\nu}\,\partial^2 u/\partial x^2} \;=\; \frac{u_0^2\,k}{u_0\,\tilde{\nu}\,k^2} \;=\; \frac{u_0}{k\,\tilde{\nu}} \tag{2.156}$$

Therefore, the viscous decay of longitudinal waves that we calculated in section 2.5.2 makes sense when the amplitude u_0 of the wave is significantly less than $k\,\tilde{\nu}$. For

example, we have the following numbers for $\tilde{\nu} = 0.225$ cm^2/s,

f	λ	$u_0 = k\,\tilde{\nu}$	pressure level
1 kHz	34 cm	0.04 cm/s	78 dB
100 kHz	0.34 cm	4.2 cm/s	118 dB

(2.157)

We calculate the pressure level in decibels using the relation (see section 2.6),

$$\text{pressure level} \;=\; 74 \;+\; 20\,\log_{10}(c_s^2\,\rho') \tag{2.158}$$

We also use the relation $u = c_s\,\rho'/\rho_0$ for a free traveling wave. The above numbers indicate that if the frequency is 1 kHz, we can neglect the nonlinear advective terms in comparison to the viscous effects, if the wave amplitude is significantly less than 0.04 cm/s. A factor of 10 would require that the pressure level is less than 58 dB, which is a very weak sound. Of course, the viscous effects increase with higher frequency, so that the viscous solution of section 2.5.2 applies to a wider range of sounds when the frequency is high.

Returning to equation 2.155 we can see that both the viscous terms and the nonlinear advective terms are smaller by a factor of u_0/c compared to the remaining terms, such as the time derivative of velocity. This is the reason why we can neglect the nonlinear and the viscous terms in many situations and work with the linear inviscid wave equation. Of course, there are limitations to the linear inviscid theory. In particular, the above comparison of equation 2.155 assumes a free traveling wave, and does not apply inside the viscous shear boundary layer where the velocity must decrease to zero in a very short distance, as we saw in section 2.5.3. In the shear boundary layer the viscous terms, such as $\nu\,\partial^2 u/\partial y^2$, are very large.

In free space the effect of nonlinearities on acoustic waves is small if the wave amplitude is much smaller than the speed of sound. We can estimate some numbers

as follows, where we use the relation $u_0/c_s = \rho'/\rho_0$ for a free traveling wave,

u_0	u_0/c_s	pressure level
34 cm/s	0.001	137 dB
344 cm/s	0.01	157 dB
3440 cm/s	0.1	177 dB

(2.159)

Therefore, nonlinear effects in free space become important for very loud sounds only.

An example of a typical nonlinear effect is frequency doubling. In particular, if we have a sinusoidal wave of frequency ω,

$$u \;=\; e^{i\,(2\,\pi\,x/\lambda \,-\, \omega\,t)} \tag{2.160}$$

the nonlinear advective term produces an oscillation of twice the original frequency,

$$u\,(\partial u/\partial x) \;=\; \frac{2\,\pi}{\lambda}\,e^{i\,(4\,\pi\,x/\lambda \,-\, 2\,\omega\,t)} \tag{2.161}$$

The frequency doubling effect is one of the reasons why the linear analysis based on complex exponentials does not work in the nonlinear regime. Of course, there are many other nonlinear effects that we do not understand, and we can not even identify them.

Although nonlinear effects are weak for acoustic waves in free space, this is not the case for acoustic waves in confined space. In particular, a wave in free space is typically generated by a small source where the acoustic energy is concentrated initially before expanding as $1/r^2$ in three-dimensional space. Therefore, near the source the wave amplitude is large, and nonlinear effects can be very important, as in the case of the air jet in a flue pipe. Nonlinear effects are the basic mechanism for amplification of sound in flue pipes (Verge94 [56], Hirschberg94 [26]).

Having discussed the limitations of linear acoustic theory in this section, the question arises whether it is reasonable to try to distinguish acoustic waves from other variations of density in nonlinear regimes. This question is important in the computer simulations of flue pipes, and is discussed below.

2.5.5 Distinguishing acoustic from hydrodynamic

In the subsonic regime, a simple rule for distinguishing acoustic waves from non-acoustic flow is the propagation speed. Acoustic waves propagate at the speed of sound which is much faster than the speed of non-acoustic or hydrodynamic flow. Hydrodynamic flow consists of vortices, boundary layers, etc that are slow-moving compared to sound waves. This difference in speed appears distinctly in the frequency domain. The frequencies of acoustic waves are typically much higher than the frequencies of non-acoustic flow, and we can exploit this property to distinguish the acoustic waves from slower hydrodynamic variations of density. In the computer simulations and in the physical experiments, a time series of the density is obtained by sampling at a fixed location in space. Then, the sound waves are identified as the relatively high frequencies in the spectrum, and hydrodynamic flow as the relatively low frequencies in the spectrum.

The above distinction between acoustic and non-acoustic motion may become blurry in regions such as near the jet of a recorder flue pipe, where acoustic waves and hydrodynamic flow interact with each other very strongly. The difficulty is that the amplitude of the acoustic motion and the amplitude of the hydrodynamic motion are comparable with each other near the jet. The oscillations of the jet generate acoustic waves and are also driven by acoustic waves, so that the two motions blur into each other and become one. This is not surprising because the acoustic and hydrodynamic regimes are simply different limits (approximations) of one flow behavior that is described by the Navier Stokes equations.

Fortunately in the case of flue pipes, the strong interactions between acoustic waves and hydrodynamic flow are limited to the region of the jet orifice and the labium (the edge which the jet impinges). Thus, a little further away from this sensitive region, the acoustic waves quickly uncouple from the slower hydrodynamic flow, and we can use the simple criterion of frequency range described above to distinguish the acoustic waves from the hydrodynamic variations of density.

temperature degrees centigrade	density 10^{-3} gm/cm^3	kinematic viscosity cm^2/s	speed of sound cm/s
15	1.226	0.145	34060
20	1.205	0.150	34290
25	1.184	0.155	34581

Table 2.2: Air-constants at various temperatures.

2.6 Appendix: units and constants

This appendix summarizes the units and constants which are employed in the computer simulations. *The speed of sound is chosen equal to* 34400 *cm/s, and the kinematic viscosity is set equal to* 0.15 cm^2/s. *These values correspond to a mean constant temperature of* 22 *degrees centigrade. Regarding the density, the units of mass are normalized by* 0.0012 gm/cm^3 *so that the mean density is unity.* Table 2.2 lists typical values of the mean density and the kinematic viscosity of air at room temperatures and atmospheric pressure. These values are taken from Newman [34, p.388]. The speed of sound of air shown in table 2.2 is calculated using the following formula from Olson [36, p.10],

$$c_s \;=\; 33100\,\sqrt{1 + 0.00366\;T} \tag{2.162}$$

where T is the temperature in degrees centigrade. The above formula for the speed of sound is equivalent to equation 2.29 which was derived in section 2.3. Note that the factor 0.00366 of equation 2.162 is equal to $1/273$, and $273 + T$ gives the absolute temperature in degrees Kelvin.

In order to compare intensities of sound, the scale of decibels of sound pressure level is used (Sekuler&Blake [45, p.298]). The scale of decibels is defined as the logarithm of the ratio of pressure fluctuation P' divided by a normalizing pressure fluctuation P'_0 which is referred to as the standard pressure level,

$$20\,\log_{10}\,\frac{P'}{P'_0} \tag{2.163}$$

The standard pressure level is the weakest sound that an average human can hear, and it is approximately,

$$P_0' \;=\; 2\times 10^{-4}\ \mathrm{gm/(cm\,s^2)} \tag{2.164}$$

Using the relation $P' = c_s^2\rho'$, the following formula is obtained,

$$\left(20\ \log_{10}\frac{P'}{P_0'}\right) \;=\; 74 \;+\; 20\ \log_{10}\left[c_s^2\,\rho_0\,\left(\frac{\rho'}{\rho_0}\right)\right] \tag{2.165}$$

The above formula is useful in the computer simulations where the normalized density fluctuations ρ'/ρ_0 appears. For the mean density of air, the value $\rho_0 = 0.0012\ \mathrm{gm/cm^3}$ is used.

It should be noted that the results of two-dimensional simulations can not be related exactly with the three-dimensional world; in particular, the two-dimensional density has units $\mathrm{gm/cm^2}$ as opposed to $\mathrm{gm/cm^3}$ for the three-dimensional density. One way of avoiding this problem of units is to work with dimensionless ratios such as ρ'/ρ_0. Of course, the problem of relating 2D to 3D results involves more than matching the units. For example, there are many 3D effects that remain un-modeled in 2D, such as 3D-expansion of waves versus a 2D-expansion, and also vortex stretching in 3D space (Tritton [54, p.114]) to mention a few (see also section 1.4 and chapter 7).

For completeness, a few definitions of dimensionless numbers are summarized here. Dimensionless numbers can be obtained by combining characteristic lengths of the flow with physical constants such as c_s and ν that appear in the Navier Stokes equations. For example, the Mach number is defined as the ratio of the flow speed divided by the speed of sound,

$$M \;=\; \frac{U}{c_s} \tag{2.166}$$

The flow speed U in the above equation is typically the maximum speed or the mean speed of the flow. In the case of subsonic jet phenomena inside flue pipes, the maximum flow speed is smaller than the speed of sound c_s by a factor of 10 to 1000, so the Mach number is between 10^{-1} and 10^{-3}.

Another important dimensionless number is the Reynolds number which measures the size of fluid inertia relative to the size of viscous effects (Tritton [54, p.97]). It is given by the ratio,

$$\mathrm{Re} \;=\; \frac{U\,l}{\nu} \tag{2.167}$$

where U and l are characteristic velocity and length scales of the flow. The choice of characteristic scales is somewhat arbitrary, and it depends on the geometry of the flow, and on which features we choose to focus on. For example, in the case of flow through a pipe (Hagen-Poiseuille flow, Landau&Lifshitz [32, p.51]), the length l is typically chosen to be the diameter of the pipe, and the speed U is chosen to be the mean speed of the flow. In the case of jets that emerge from a narrow orifice, such as the ones of section 1.4 and chapter 7, the convention is also to choose l as the diameter of the orifice, and U the mean speed of the flow.

A third dimensionless number that is relevant in simulations of subsonic jets is the Strouhal number, which measures the relative frequency of oscillation. For example, if a jet executes transverse oscillations relative to its forward motion, then the Strouhal number can be defined as the ratio of the frequency f of oscillations multiplied by the diameter l of the jet, and divided by the jet speed U,

$$\mathrm{St} \;=\; \frac{f\,l}{U} \tag{2.168}$$

Other dimensionless numbers in addition to the above can be found in standard textbooks (Batchelor [3] and Newman [34]) and in specialized areas of fluid mechanics.

The next chapter begins the discussion of numerical methods.

Chapter 3

Numerical methods for fluid flow

Except for special cases, the Navier Stokes equations of the previous chapter can not be solved analytically. Therefore, numerical methods must be used. Below, the basic ideas of finite difference methods are reviewed. Subsequently, an explicit finite difference method for solving the compressible Navier Stokes equations is described. Also, an explicit finite difference method for solving the incompressible Navier Stokes equations is described which is used for numerical testing purposes only. Most of the ideas presented here can be found in textbooks of computational fluid dynamics. Some results which are not easily available in the literature (as far as I know) are the discussion on why *explicit numerical methods* are appropriate for subsonic flow in section 3.2.1, and the analysis of the CFL (Courant-Friedrichs-Lewy) condition in section 3.3.2.

3.1 Numerical grids

The Navier Stokes equations can be solved numerically by introducing a numerical grid in space and time. For the sake of simplicity, only the spatial dimensions of the grid are described here. To include a time dimension, we can imagine making copies of the planar grids shown in figure 3-1, and stacking them on top of each other.

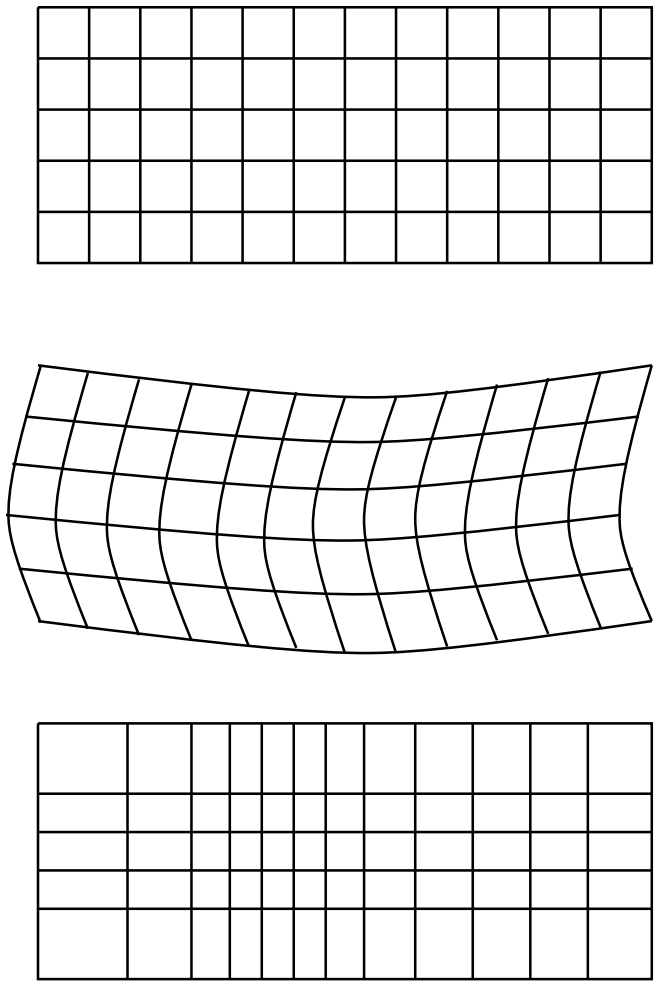

Figure 3-1: Three simple types of numerical grids: uniform orthogonal, curvilinear, non-uniform orthogonal.

A numerical grid defines a discretization of spacetime, and replaces the continuous functions of density and velocity ρ, V_x, V_y by a discrete set of values defined at the nodes of the grid (namely, the points where the grid lines of figure 3-1 intersect).

Numerical grids are distinguished into staggered or non-staggered depending on whether the fluid variables are defined exactly at the grid nodes, or halfway between the grid nodes. For example, the fluid velocity can be defined halfway between the grid nodes, and the fluid density can be defined exactly at the grid nodes. This staggered allocation of variables has advantages in some cases (Peyret&Taylor [38]),

but it is more complex to deal with than a straightforward non-staggered grid where all the variables are defined at the grid nodes. In the present work, non-staggered grids are employed exclusively.

Numerical grids can be distinguished into uniform or non-uniform. For example, the grid shown at the top of figure 3-1 is a uniform orthogonal grid, and the grid used in section 4.1.1 is a uniform hexagonal grid. In the present work, only uniform grids are used because they are very simple to program and highly-suited for parallel computing. Another reason for using uniform grids is that the lattice Boltzmann method works only with uniform grids as far as is known today. The only way to extend lattice Boltzmann to non-uniform grids is to employ two grids of different uniform resolution joined together via interpolation (a technique called composite grids). This idea for lattice Boltzmann is outlined in section 4.6.2.

Non-uniform grids increase the resolution (density of grid points) in certain regions, while decreasing the resolution in other regions where the flow is smooth and not much is happening. Sometimes, the change of resolution introduces numerical artifacts. To minimize the artifacts, the resolution of the grid should be varied smoothly, if possible. On the other hand, smoothness does not guarantee the absence of artifacts. In particular, acoustic waves are very sensitive to changes of resolution, and should be carefully tested in regions where the resolution is changing.

Non-uniform grids include the composite grids mentioned above, and also curvilinear and non-uniform orthogonal grids which are shown at the middle and bottom of figure 3-1. In the case of curvilinear grids, a coordinate transformation from the curvilinear space to a uniform orthogonal space (called logical space) is usually employed. The Navier Stokes equations are transformed to new coordinates, and finite differences are applied to the transformed equations on the logical grid [52]. Curvilinear grids are often designed to be body-conforming in order to approximate closely the shape of smooth boundaries such as airfoils.

An alternative to coordinate transformations is to discretize the Navier Stokes

equations directly in the physical space by taking finite differences based on the local spacings around each grid point (Peyret&Taylor [38, p.326]). Direct discretization in the physical space is typically used in the case of non-uniform orthogonal grids, and also in the case of unstructured non-uniform grids described below.

Both the curvilinear and the non-uniform orthogonal grids of figure 3-1 are well-structured grids. By contrast, there are also unstructured non-uniform grids (not shown here) where the grid points are "layed-out" with almost complete freedom in order to match the boundaries and the areas where higher resolution is needed (Camp&et al. [6]). Unstructured grids are very popular and very promising. A lot of research is currently being done to find good ways of parallelizing unstructured grids.

The above catalogue of numerical grids should put in perspective the uniform grids which are used here. Uniform grids are not the most efficient grids that are possible, but they are very simple to use, and very easy to parallelize. In the next section, the choice between explicit and implicit methods is discussed.

3.2 Explicit versus implicit

Numerical methods for fluid dynamics can be distinguished into explicit and implicit. In the case of an explicit method, the future value $V(t + \Delta t, x, y)$ of a fluid variable $V(t, x, y)$ at the grid point (x, y) depends only on the present and past values at neighboring points. In other words, an explicit method uses only local interactions to calculate the future values of density and velocity. An example of an explicit method is shown graphically at the left side of figure 3-2 with the time axis increasing in the vertical direction. Here, the future value of the central node depends on the present value of the central node and also on the present values of the four neighbors.

In the case of an implicit method, the future value $V(t+\Delta t, x, y)$ of a fluid variable $V(t, x, y)$ at the grid point (x, y) depends on the future values of neighboring nodes such as $V(t + \Delta t, x + \Delta x, y + \Delta y)$ (right side of figure 3-2). This implies that the

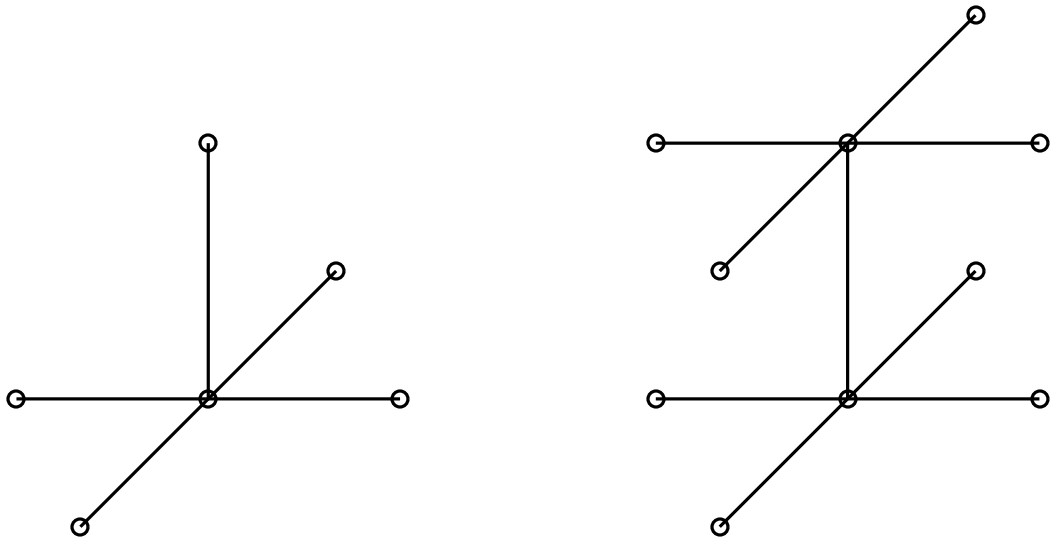

Figure 3-2: Explicit and implicit discretizations in two-dimensional space with the time axis increasing vertically.

neighboring node $V(t+\Delta t, x+\Delta x, y+\Delta y)$ depends on other nodes further down the grid in a similar way, for example $V(t+\Delta t, x+2\Delta x, y+2\Delta y)$. Consequently, an implicit method couples together distant nodes, and introduces a large matrix equation that extends the length of the numerical grid. The matrix equation makes implicit methods more stable than explicit methods, but also more difficult to parallelize.

Explicit methods are very simple, ideally scalable, and highly suitable for large parallel computers with small communication capabilities (see chapter 6). However, explicit methods require small integration time steps in order to remain numerically stable. By contrast, implicit methods are challenging to parallelize, and have large communication requirements. However, implicit methods can use much larger integration time steps than explicit methods. Because of these differences between explicit and implicit methods, the decision of which method to use depends on the available computer system and on the problem's requirements regarding the integration time step. For instance, the simulation of subsonic flow requires small integration time steps in order to follow the fast-moving acoustic waves (see below). Thus, an explicit method is generally a good choice for subsonic flow.

A possibility that deserves to be explored in the future is intermediate methods between explicit and implicit. By this, I do not mean semi-implicit methods where some terms of the Navier-Stokes equations are discretized implicitly and others explicitly. Also, I do not mean "alternating directions" (Peyret&Taylor [38]) where the matrix equation is split into smaller matrices that are solved in succession: first along the x-direction, then along the y-direction, then along the z-direction. Such approaches reduce the size of the matrix that accompanies an implicit method, but still produce a matrix that extends the length of the numerical grid, and presents formidable difficulties for parallel computing. Instead, the real breakthrough would be to develop numerical methods that have the stability properties of implicit methods without using matrices that extend the whole grid. Such "intermediate" methods would retain some of the locality of explicit methods that is very important for parallel computing. An effort towards this direction in the context of the diffusion equation is discussed in [2] and references therein.

3.2.1 Small integration time steps for subsonic flow

The integration time step Δt in simulations of subsonic flow must be small both for explicit and implicit methods. An approximate constraint on the numerical speed $\Delta x/\Delta t$ of explicit methods can be obtained from the CFL condition (Courant-Friedrichs-Lewy) which says that *the domain of numerical dependence must include the domain of physical dependence.* The CFL condition must be satisfied in order to be able to simulate the physical phenomenon. In the case of simple hyperbolic problems (such as the wave equation), it can be shown that the CFL condition is also a necessary condition for the stability of explicit methods (Courant&et al. [14]). The CFL condition can be written approximately as follows,

$$\frac{\Delta x}{\Delta t} \geq c_s \tag{3.1}$$

where c_s is the propagation speed of acoustic waves. In other words, the CFL condition requires that the numerical speed $\Delta x/\Delta t$ must be at least as large as the physical speed. A more accurate formula for the CFL condition is derived in sections 3.3.1 and 3.3.2.

In the case of implicit methods, the CFL condition can not be applied directly because the matrix of an implicit method introduces dependencies (interactions) between distant nodes along the entire length of the numerical grid. Therefore, the numerical speed of an implicit method is, in some sense, the length the grid divided by Δt, which is a very large numerical speed. On the other hand, this speed can not be compared with the physical speed of acoustic waves in a meaningful way because the matrix-introduced interactions are not physical interactions. Another difficulty in trying to interpret the CFL condition in the context of implicit methods is that many implicit methods are known to be unconditionally stable (Peyret&Taylor [38]) under linear stability analysis. Therefore, such methods can compute a stable solution (though not necessarily accurate or correct) even when the time step Δt is much larger than the CFL limit. All this shows that the CFL condition is inconclusive in the case of implicit methods.

An approximate constraint on the numerical speed $\Delta x/\Delta t$ of implicit methods can be obtained by inquiring whether the computed solution simulates accurately the physical phenomena under consideration. In the case of acoustic waves that propagate through the fluid and reflect off obstacles, the time step Δt must be small enough to follow the propagation of acoustic waves. In particular, the product $c_s\,\Delta t$ must be less than a few Δx in order to have enough resolution to simulate the passage and reflection of acoustic waves,

$$\Delta x \;\sim\; c_s \Delta t \tag{3.2}$$

The above constraint arises from the time-scales of the problem, and applies both to implicit and explicit methods.

As stated earlier, throughout this work only explicit methods are used. In the next

section, an explicit finite difference method is described for solving the compressible Navier Stokes equations.

3.3 Compressible finite difference method

Let us consider a uniform orthogonal grid with $\Delta x, \Delta y, \Delta z, \Delta t$ intervals in space and time. For the sake of brevity, only two spatial dimensions are shown here. The extension of the method to three dimensions is straightforward. The following abbreviated notation is used,

$$\rho^n_{j,k} = \rho(x_0 + j\Delta x, y_0 + k\Delta y, t_0 + n\Delta t) \tag{3.3}$$

where x_0, y_0 denote the space coordinates of the point at the left-bottom corner of the grid according to a Cartesian coordinate system, and t_0 is the starting time of the integration. Below, variables without any space sub-indices, for example ρ^{n+1}, are assumed to be $\rho^{n+1}_{j,k}$. Also, the notation $u = V_x$ and $v = V_y$ is used to avoid confusion with indices. The continuous Navier-Stokes equations in two dimensions can be written as follows,

$$\frac{\partial \rho}{\partial t} + \frac{\partial(\rho u)}{\partial x} + \frac{\partial(\rho v)}{\partial y} = 0 \tag{3.4}$$

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial v}{\partial y} + c_s^2\frac{\partial \rho}{\rho \partial x} - \nu\nabla^2 u = 0 \tag{3.5}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + c_s^2\frac{\partial \rho}{\rho \partial y} - \nu\nabla^2 v = 0 \tag{3.6}$$

If we use the following difference operators (forward-Euler for time and symmetric differences for space),

$$\frac{\partial u}{\partial t} \quad \rightarrow \quad \delta_t u = \frac{u^{n+1} - u^n}{\Delta t} \tag{3.7}$$

$$\frac{\partial u}{\partial x} \quad \rightarrow \quad \delta_x u_{j,k} = \frac{u_{j+1,k} - u_{j-1,k}}{2\,\Delta x} \tag{3.8}$$

$$\frac{\partial u}{\partial y} \quad \rightarrow \quad \delta_y u_{j,k} = \frac{u_{j,k+1} - u_{j,k-1}}{2\,\Delta y} \tag{3.9}$$

$$\frac{\partial^2 u}{\partial x^2} \quad \rightarrow \quad \delta_{xx} u_{j,k} = \frac{u_{j+1,k} - 2u_{j,k} + u_{j-1,k}}{\Delta x^2} \tag{3.10}$$

the discretized Navier-Stokes equations can be written as follows,

$$\rho^{n+1} = \rho^n - \Delta t \left[\rho^n \delta_x u^{n+1}_{j,k} + \rho^n \delta_y v^{n+1}_{j,k} + u^{n+1}_{j,k} \delta_x \rho^n_{j,k} + v^{n+1}_{j,k} \delta_y \rho^n_{j,k} \right] \tag{3.11}$$

$$u^{n+1} = u^n + \Delta t \left[\nu \, \rho^n (\delta_{xx} u^n_{j,k} + \delta_{yy} u^n_{j,k}) - \frac{c_s^2}{\rho} \delta_x \rho^n_{j,k} - u^n \delta_x u^n_{j,k} - v^n \delta_y u^n_{j,k} \right] \tag{3.12}$$

$$v^{n+1} = v^n + \Delta t \left[\nu \, \rho^n (\delta_{xx} v^n_{j,k} + \delta_{yy} v^n_{j,k}) - \frac{c_s^2}{\rho} \delta_y \rho^n_{j,k} - u^n \delta_x v^n_{j,k} - v^n \delta_y v^n_{j,k} \right] \tag{3.13}$$

Equations 3.12 and 3.13 produce immediately the new velocity at the next time step $t + \Delta t$ because all the terms of the momentum equations are discretized explicitly (evaluated at time t). Equation 3.11 however is slightly different from the momentum equations. The mass continuity equation is discretized in a semi-implicit way which means that the velocity values at time $t + \Delta t$ are used to compute the new density value $\rho(t + \Delta t)$ at time $t + \Delta t$. In other words, the computation proceeds in two steps: First, the new velocity is calculated, and then the new density is calculated in a separate loop. This two-step procedure is very important for numerical stability. If both the density and the velocity are discretized explicitly, the algebraic system becomes very unstable. This can be easily checked in numerical experiments, and a plausible theoretical explanation is given in section 3.3.3.

3.3.1 Numerical stability

Numerical stability conditions for the explicit finite difference method (3.11 to 3.13) are not known exactly. However, a few approximate estimates can be obtained. First, the CFL condition says that the domain of numerical dependence must include the domain of physical dependence. After some manipulations, the following conditions are obtained (see section 3.3.2 for a detailed derivation),

$$\frac{\Delta x}{\Delta t} \geq (|V_x| + |V_y| + c_s \sqrt{2}) \tag{3.14}$$

or more generally,

$$\Delta t \leq \left(\frac{|V_x|}{\Delta x} + \frac{|V_y|}{\Delta y} + c_s \sqrt{\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}} \right)^{-1} \tag{3.15}$$

To satisfy the above equations, the time step of integration Δt must be kept very small in the case of subsonic flow where the speed of sound c_s is very large.

Another stability condition that takes into account viscous effects can be derived as follows. We consider the linear advection-diffusion equation (Peyret&Taylor [38, p.65]) which is simple to analyze, and is a special case of the momentum Navier Stokes equations. The advection-diffusion equation has the following form,

$$\frac{\partial f}{\partial t} + A\frac{\partial f}{\partial x} + B\frac{\partial f}{\partial y} - \nu \nabla^2 f = 0 \tag{3.16}$$

where f is the variable that is diffused; for example, the fluid momentum. The coefficients A and B correspond to the fluid speed

$$A = |V_x|$$

$$B = |V_y|$$

and they are assumed to be constant for the purpose of linear analysis. The explicit discretization of equation 3.16 produces,

$$f^{n+1} = f^n - \Delta t \left(A\delta_x f^n + B\delta_y f^n - \nu \delta_{xx} f^n - \nu \delta_{yy} f^n \right) \tag{3.17}$$

By applying the von Neumann stability analysis to the above (see section 3.3.3 for a description), we get the following constraints (Peyret&Taylor [38, p.65]) in the case of $\Delta x = \Delta y$,

$$\Delta t \leq \frac{2\nu}{|V_x|^2 + |V_y|^2} \tag{3.18}$$

and also,

$$\frac{\nu \Delta t}{\Delta x^2} \leq \frac{1}{4} \tag{3.19}$$

Although the above conditions are necessary, they are not sufficient. The simulation of subsonic compressible flow at high Reynolds numbers is susceptible to slow-growing

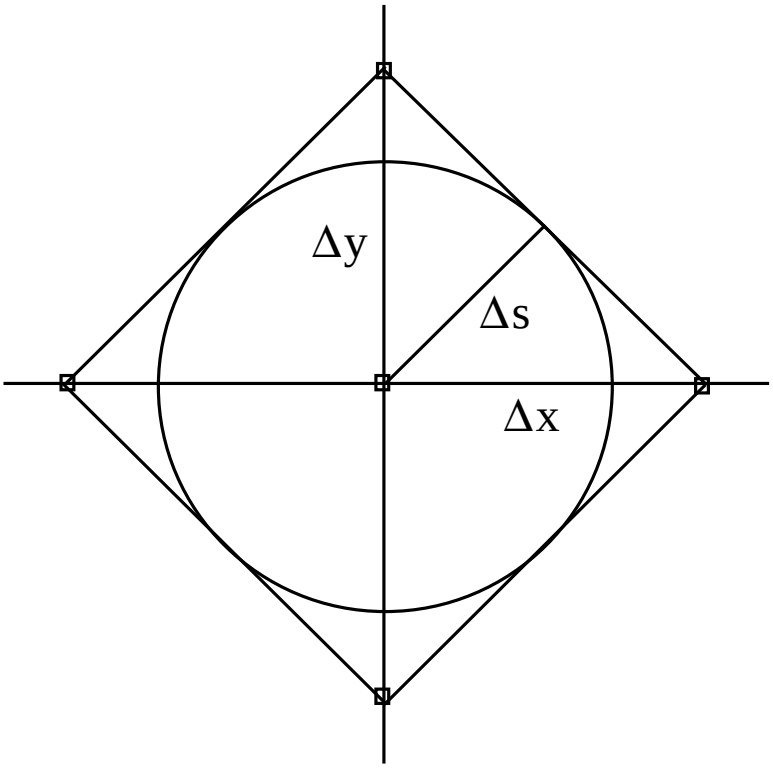

Figure 3-3: The numerical domain of dependence for $\Delta x = \Delta y$.

numerical instabilities of very high spatial frequency. These stability problems are discussed in chapter 5, and they can be avoided by including artificial viscosity (fourth-order numerical dissipation) which filters very high spatial frequencies.

3.3.2 Derivation of CFL formula

To derive the CFL stability equation 3.14, we consider a node with four neighbors in a square grid as shown in figure 3-3. The goal is to compare the numerical and the physical domains of dependence. We observe that the four neighbors are the only nodes that can influence the central node after one time step Δt. Thus, the numerical domain of dependence of the central node is the square area that is enclosed by straight lines drawn between the four neighbors. The physical domain of dependence that arises from acoustic waves is a circle of radius $\Delta s = c_s \Delta t$, and it must be contained

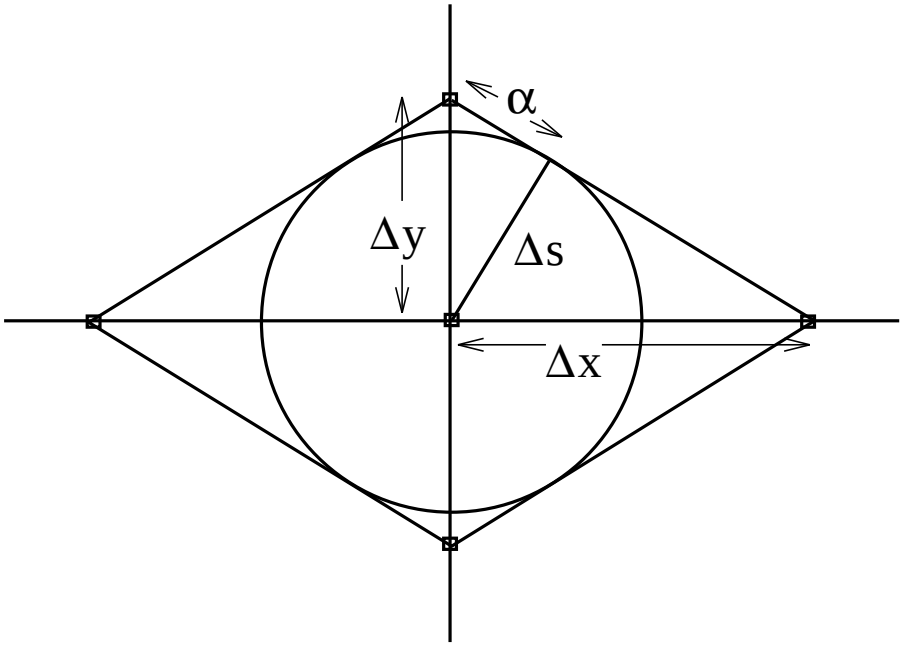

Figure 3-4: The numerical domain of dependence for $\Delta x \neq \Delta y$.

within the square area. Simple geometry shows that the maximum radius Δs is given by the following formula,

$$\Delta s = \frac{1}{2}(\sqrt{2}\Delta x) \tag{3.20}$$

and thus we must have,

$$c_s \Delta t \leq \frac{1}{2}(\sqrt{2}\Delta x) \tag{3.21}$$

$$\frac{\Delta x}{\Delta t} \geq c_s \sqrt{2} \tag{3.22}$$

Similarly, the physical domain of dependence that arises from hydrodynamic motion must be contained within the numerical domain. Thus, we must have,

$$\frac{\Delta x}{\Delta t} \geq |V_x| \tag{3.23}$$

$$\frac{\Delta x}{\Delta t} \geq |V_y| \tag{3.24}$$

A simple way to combine all of the above inequalities is to require that $\Delta x/\Delta t$ is greater than the sum of the individual positive terms. This produces the inequality

that we wished to prove,

$$\frac{\Delta x}{\Delta t} \geq (|V_x| + |V_y| + c_s\sqrt{2}) \tag{3.25}$$

The more general CFL stability equation 3.15 is derived in a similar way. We consider the numerical domain shown in figure 3-4 that has a rhombic shape. We have the following geometric relations, where α is the length shown in figure 3-4,

$$\Delta s^2 + \alpha^2 = \Delta y^2 \tag{3.26}$$

$$\Delta s^2 + \left(\sqrt{\Delta x^2 + \Delta y^2} - \alpha\right)^2 = \Delta x^2 \tag{3.27}$$

After some algebra we can obtain,

$$\Delta s = \left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}\right)^{-1} \tag{3.28}$$

The physical domain of dependence that arises from acoustic waves must be smaller than the numerical domain. Thus, we must have,

$$\Delta s \geq c_s \Delta t \tag{3.29}$$

or equivalently,

$$\Delta t^{-1} \geq c_s \left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}\right) \tag{3.30}$$

We must also satisfy the hydrodynamic constraints,

$$\Delta t^{-1} \geq \frac{|V_x|}{\Delta x} \tag{3.31}$$

$$\Delta t^{-1} \geq \frac{|V_y|}{\Delta y} \tag{3.32}$$

The above inequalities can be combined additively to produce the inequality,

$$\Delta t \leq \left(\frac{|V_x|}{\Delta x} + \frac{|V_y|}{\Delta y} + c_s\sqrt{\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}}\right)^{-1} \tag{3.33}$$

This is the general form of the CFL condition in two dimensions for an explicit numerical method that employs nearest neighbor interactions.

3.3.3 Semi-implicit density

An explanation why a semi-implicit discretization of the continuity equation leads to better stability properties than a fully explicit discretization is as follows. Let us write the discretized Navier-Stokes equations in one-dimensional form for simplicity. We write the mass continuity equation and the momentum conservation equation along the x-direction as follows,

$$\rho^{n+1} = \rho^n - \Delta t \left[\rho^n \frac{u^{n+1}_{j+1} - u^{n+1}_{j-1}}{2\Delta x} + u^{n+1} \frac{\rho^n_{j+1} - \rho^n_{j-1}}{2\Delta x} \right] \tag{3.34}$$

$$u^{n+1} = u^n + \Delta t \left[\nu \rho^n \frac{u^n_{j+1} - 2u^n + u^n_{j-1}}{\Delta x^2} - c_s^2 \frac{\rho^n_{j+1} - \rho^n_{j-1}}{\rho^n 2\Delta x} - u^n \frac{u^n_{j+1} - u^n_{j-1}}{2\Delta x} \right] \tag{3.35}$$

Equation 3.34 is a semi-implicit discretization of the continuity equation. To compare, an explicit discretization is as follows,

$$\rho^{n+1} = \rho^n - \Delta t \left[\rho^n \frac{u^n_{j+1} - u^n_{j-1}}{2\Delta x} + u^n \frac{\rho^n_{j+1} - \rho^n_{j-1}}{2\Delta x} \right] \tag{3.36}$$

We now apply the von-Neumann frequency analysis (Peyret&Taylor [38, p.344]). We write the different variables in terms of their frequency components, and we analyze each frequency separately (non-linear combinations of frequencies are ignored). We have,

$$\begin{aligned} \rho^n &= e^{i\kappa_0 x} \\ \rho^{n+1} &= G_0\, e^{i\kappa_0 x} \\ u^n &= A\, e^{i\kappa_1 x} \\ u^{n+1} &= G_1\, A\, e^{i\kappa_1 x} \end{aligned} \tag{3.37}$$

where A is the velocity amplitude, and G_0, G_1 are the growth factors corresponding to the spatial frequencies κ_0, κ_1 of the density and velocity respectively. The imaginary unit of complex numbers is denoted by $i = \sqrt{-1}$, and it should not be confused with indices because i is never used as an index here. The following identities are very useful,

$$\begin{aligned} \delta_x \rho^n &= \rho^n\, i\Delta x^{-1} \sin(\kappa_0 \Delta x) \\ \delta_{xx} \rho^n &= \rho^n\, 2\,\Delta x^{-2} (\cos(\kappa_0 \Delta x) - 1) \end{aligned} \tag{3.38}$$

Below, the following dimensionless constants are used for brevity,

$$\begin{aligned} \zeta_1 &= (\Delta t/\Delta x)A \\ \zeta_2 &= (\nu \Delta t/\Delta x^2) \\ \zeta_3 &= (\Delta t/\Delta x)(c_s^2/A) \end{aligned} \tag{3.39}$$

If we substitute the exponentials of equation 3.37 into equation 3.34 and 3.35, we obtain the following equations,

$$G_0 \;=\; 1 - i\, G_1 \zeta_1\, e^{i\kappa_1 x}(\sin\kappa_0\Delta x + \sin\kappa_1\Delta x) \tag{3.40}$$

$$G_1 \;=\; 1 + 2\zeta_2(\cos\kappa_1\Delta x - 1) - i\,\zeta_3\, e^{-i\kappa_1 x}\sin\kappa_0\Delta x - i\,\zeta_1\, e^{i\kappa_1 x}\sin\kappa_1\Delta x \tag{3.41}$$

By contrast, the explicit discretization of the continuity equation produces the following,

$$G_0 \;=\; 1 - i\,\zeta_1\, e^{i\kappa_1 x}(\sin\kappa_0\Delta x + \sin\kappa_1\Delta x) \tag{3.42}$$

A necessary condition for stability is that the magnitude of each growth factor individually G_0, G_1 should not be larger than unity for all possible frequencies κ_0, κ_1. The largest frequency that is possible on a grid of spacing Δx corresponds to a wavelength of $2\Delta x$ (2 nodes per cycle),

$$0 \le \kappa_0, \kappa_1 \le \frac{\pi}{\Delta x} \tag{3.43}$$

Different choices of κ_0, κ_1 within the above range can be substituted in equations 3.40, 3.41, and 3.42, 3.41 to derive stability conditions. The algebra is rather complicated, and is omitted here. Instead, we notice that the G_0 factor of the semi-implicit version (equation 3.40) is almost identical to the G_0 factor of the explicit version (equation 3.42) except for the extra G_1. In the explicit version, the magnitude of G_0 is always greater than unity, but in the semi-implicit version the magnitude of G_0 can be less than unity because of the extra G_1. A complete analysis requires carrying out the complex multiplications, collecting terms, considering the variation of $e^{i\kappa_1 x}$ in space, etc. The above preliminary analysis gives a basic idea of why the semi-implicit version can be expected to be more stable than the explicit version, a fact which can be easily observed experimentally.

3.3.4 Boundary conditions

The modeling of boundaries is a very important part of a numerical method. The boundaries include the internal obstacles and the perimeter that encloses the simulated region (it should be noted that periodic boundaries are not useful in the case of flue pipes). Near a boundary, the numerical method must take into account the fact that *grid points are available only on the interior side of the boundary.* For instance, the symmetric differences which are used at the interior nodes (equation 3.8) must be replaced with asymmetric differences at the boundary nodes. Furthermore, the numerical boundary conditions must be chosen properly to model the desired physical conditions such as a non-slip wall, an inlet, and an outlet.

A non-slip wall means that the velocity variables V_x, V_y are always equal to zero; therefore, only the density needs to be calculated at a non-slip wall. The approach which is used in the simulations of flue pipes, is to compute the density ρ by applying asymmetric finite differences to the continuity equation. In particular, the central differences of equation 3.8 are replaced with asymmetric differences denoted by δ_{x^-} and δ_{x^+} as follows,

$$\frac{\partial \rho}{\partial x} \quad \rightarrow \quad \delta_{x^-}\rho = \frac{3\rho_{j,k} - 4\rho_{j-1,k} + \rho_{j-2,k}}{2\,\Delta x} \tag{3.44}$$

$$\frac{\partial \rho}{\partial x} \quad \rightarrow \quad \delta_{x^+}\rho = \frac{-3\rho_{j,k} + 4\rho_{j+1,k} - \rho_{j+2,k}}{2\,\Delta x} \tag{3.45}$$

and similarly for the y-directions.

An alternative approach, which is not used in the simulations of flue pipes, is to compute the density at a non-slip wall by simple extrapolation in a normal direction to the boundary wall. Preliminary experiments which I have performed, indicate that in the case of non-slip walls, the continuity equation with asymmetric differences works better than extrapolating the density. However, the extrapolation approach is described here for completeness. Extrapolation amounts to setting

$$\rho(x_B) = \rho(x_B - \Delta x) \tag{3.46}$$

where x_B is the boundary wall. The justification for the extrapolation condition comes from considering the momentum Navier Stokes equation at the wall,

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial v}{\partial y} + c_s^2\frac{\partial \rho}{\rho \partial x} - \nu\nabla^2 u = 0 \tag{3.47}$$

Since $u = v = 0$, most of the above terms vanish, and we obtain,

$$c_s^2\frac{\partial \rho}{\rho \partial x} - \nu\frac{\partial^2 u}{\partial x^2} = 0 \tag{3.48}$$

The speed of sound c_s is very large compared to the flow speed u. Thus, it makes sense to approximate the above with the condition $\partial\rho/\partial x = 0$ which gives the extrapolation condition for the density at a non-slip wall.

There are also other approaches for calculating the density at the boundary (more sophisticated than the above), and some of them are described in Poinsot&Lele [39]. After some algebra, it is possible to show that the formulas of Poinsot&Lele [39] in the case of a non-slip wall are equivalent to applying asymmetric differences to the continuity equation with the addition of some correction terms which are proportional to the Mach number; hence, they are small in the case of subsonic flow. Because the correction terms introduce complexity and additional finite differencing, I do not use them in the simulations of flue pipes.

Boundary conditions for modeling an inlet and an outlet are discussed later in section 7.3. Below, a finite difference method for simulating incompressible flow is described.

3.4 Incompressible finite difference method

The incompressible finite difference method described here, is employed for numerical testing purposes only. In particular, in sections 4.4 and 4.5 the numerical accuracy of the lattice Boltzmann method is tested on fluid flows that have exact analytic solutions. These exact solutions assume a perfectly incompressible flow, and they ignore acoustic waves. To compare the lattice Boltzmann method with methods

specifically designed for perfectly incompressible flows, the following incompressible finite difference method is used. The continuity equation 2.52 is replaced with the divergence-free condition for the velocity field,

$$\frac{\partial V_x}{\partial x} + \frac{\partial V_y}{\partial y} + \frac{\partial V_z}{\partial z} = 0 \tag{3.49}$$

The momentum equations remain as before, namely,

$$\frac{\partial V_x}{\partial t} + V_x\frac{\partial V_x}{\partial x} + V_y\frac{\partial V_x}{\partial y} + V_z\frac{\partial V_x}{\partial z} + \frac{\partial P}{\partial x} - \nu\nabla^2 V_x = 0 \tag{3.50}$$

$$\frac{\partial V_y}{\partial t} + V_x\frac{\partial V_y}{\partial x} + V_y\frac{\partial V_y}{\partial y} + V_z\frac{\partial V_y}{\partial z} + \frac{\partial P}{\partial y} - \nu\nabla^2 V_y = 0 \tag{3.51}$$

To advance the solution, the momentum equations are discretized explicitly; while the pressure term is omitted when calculating the first estimate of the velocity. Then, the velocity estimate is corrected in order to satisfy incompressibility by solving a Poisson equation,

$$\nabla^2\phi = \frac{\partial V_j^*}{\partial x_j} \tag{3.52}$$

where V_j^* is the first estimate of the velocity, and the Einstein summation is implied. The above Poisson equation computes the part of the velocity field that has non-zero divergence, which is then subtracted from the initial velocity estimate to obtain a divergent-free velocity as follows,

$$V_i(t+\Delta t) = V_i^* - \frac{\partial\phi}{\partial x_i} \tag{3.53}$$

The correction of the velocity can also be view as a projection of the initial velocity field onto the space of divergent-free velocity fields. Accordingly, this method is called a *projection method*. The projection takes into account the pressure effects that were omitted in the first estimate of the velocity (Peyret&Taylor [38, p.160]). In addition, the solution of the Poisson equation provides an estimate of the pressure at the current time-step as follows,

$$P = \frac{1}{\Delta t}\,\phi \tag{3.54}$$

In the numerical tests of sections 4.4 and 4.5, the Poisson equation is solved with Successive-Over-Relaxation (SOR) [40, page 680] using an orthogonal non-staggered grid. Also, forward-Euler is used to estimate the time derivative, and centered differences (3-point symmetric) are used to calculate the spatial derivatives.

In the next chapter, the lattice Boltzmann method for simulating subsonic compressible flow is presented.

Chapter 4

The lattice Boltzmann method

The lattice Boltzmann (LB) method is a numerical scheme for simulating viscous compressible flow in the subsonic regime (Koelman [29], Qian [41], Chen [10]). In this chapter, the LB method is analyzed, and two major results are presented: the development of a new technique for accurate boundary and initial conditions for the LB method, and the demonstration that the LB method is second-order accurate in space and in time.

In the next section, the basic LB algorithm is reviewed, and the hexagonal 7-speed LB model is described. The 7-speed model has the smallest number of populations F_i that are necessary to give correct Navier Stokes in two dimensions. Because of its simplicity, the 7-speed model is used in all the theoretical discussions here. In section 4.2, techniques for accurate boundary and initial conditions for the LB method are analyzed. In section 4.3, the 9-speed LB model for 2D orthogonal grids, and also the 15-speed LB model for 3D orthogonal grids are described.

In sections 4.4 and 4.5, the numerical accuracy of the LB method is tested experimentally on initial and on boundary value problems. The LB method is shown to be second-order accurate in space and in time. Also, the LB method is compared against an explicit finite difference method for incompressible flow. In section 4.6.1, the modeling of non-slip wall and the calculation of density at a non-slip wall are dis-

cussed. In section 4.6.2, an approach for developing composite grids (grids of different resolution joined together) for the LB method is outlined.

There is also an appendix where the numerical roundoff error of the LB method is analyzed (section 4.7.1), and the relationship between lattice gas and lattice Boltzmann is discussed (section 4.7.2).

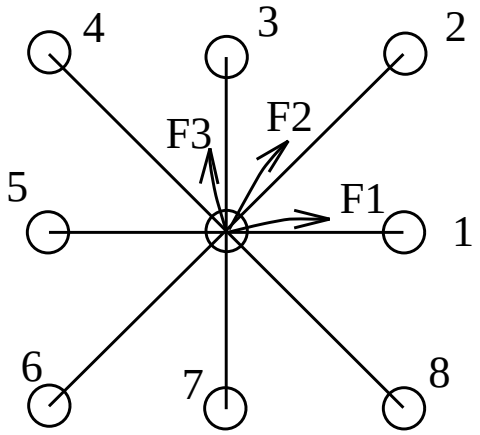

Figure 4-1: The 8 moving populations of the orthogonal lattice Boltzmann method.

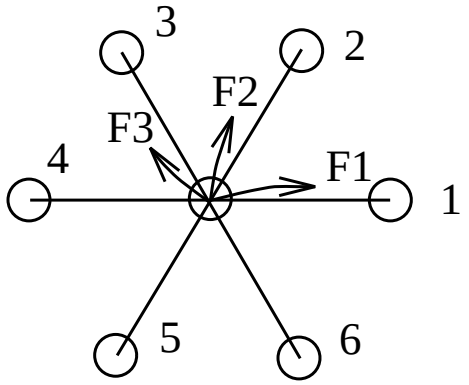

Figure 4-2: The 6 moving populations of the hexagonal lattice Boltzmann method.

4.1 Basics of lattice Boltzmann

The ideas behind the lattice Boltzmann approach (and lattice gas of section 4.7.2) come from the kinetic theory of gases. According to kinetic theory, the dynamics of flows at length scales comparable to the mean free path are described by a Boltzmann equation,

$$\frac{\partial f}{\partial t} + \vec{v} \cdot \vec{\nabla} f \;=\; (-1/\tau)(f - f^{\mathrm{eq}}) \tag{4.1}$$

where $f(\vec{x}, \vec{v}, t)$ represents the density of particles inside an infinitesimal volume $(\vec{x}, \vec{x}+d\vec{x})$ with velocity $(\vec{v}, \vec{v}+d\vec{v})$ at time t. The left-hand side of equation 4.1 represents the advection of particles with velocity $\vec{v}$, and the right-hand side represents the collision between particles. The collision operator of equation 4.1 is known as the BGK [4] relaxation with time constant τ towards local equilibrium f^{eq} (typically, a Maxwell-Boltzmann equilibrium).[1] Starting from the Boltzmann equation 4.1 which describes flow at microscopic scales, it is possible to derive the Navier Stokes equations which describe flow at macroscopic scales (at least 100 times the mean free path). Such a derivation requires a suitable averaging of the Boltzmann equation over all possible velocities, and also a Chapman-Enskog expansion (see section 4.1.2).

The lattice Boltzmann method takes the structure of the Boltzmann equation 4.1 and the ideas of kinetic theory, and applies them to macroscopic length scales using a discrete set of velocities instead of a continuous set of velocities $\vec{v}$. Despite the coarsening of length scale and the discrete set of velocities, the lattice Boltzmann method manages to produce the Navier Stokes equations in a similar way that kinetic theory does. The key ingredients that make the kinetic approach work, are the advection of particles and the collision of particles (relaxation) conserving mass, momentum, and energy. An additional feature is that the discrete set of velocities requires a highly-symmetric lattice (grid) on which the particles can move [20, 15, 58].

In two dimensions, typical lattices are the hexagonal and the orthogonal lattices

[1] More complex collision operators can be used also to describe 2-pair, 3-pair, etc interactions between particles (this leads to the BBGKY hierarchy of equations [27, p.65]).

shown in figures 4-1 and 4-2. In the lattice Boltzmann method, each node of the lattice is associated with a set of moving particles or populations F_i. The fluid variables ρ, V_x, V_y can be obtained from the F_i via a simple summation at each fluid node,

$$\begin{aligned} \rho &= \sum F_i \\ \rho \vec{V} &= \sum F_i \vec{e}_i \end{aligned} \tag{4.2}$$

where $\vec{e}_i$ are the discrete velocities of the lattice; for example, on a hexagonal lattice,

$$\vec{e}_i = \frac{\Delta x}{\Delta t} \left(\cos \frac{2\pi(i-1)}{6}, \ \sin \frac{2\pi(i-1)}{6} \right) \tag{4.3}$$

where Δx is the lattice spacing (distance between neighboring nodes) and Δt is the integration time step. Each node in a hexagonal lattice has 6 nearest neighbors, and the simplest lattice Boltzmann method has 6 moving populations at each node. These populations are shifted (advected) from one lattice site to another, and are relaxed towards local equilibrium by means of a collision operator which conserves mass, momentum, and energy just like a particle collision. The evolution equation is as follows,

$$F_i(\vec{x} + \vec{e}_i \, \Delta t, \, t + \Delta t) \ = \ F_i(\vec{x}, t) \ + \ C_i \tag{4.4}$$

where C_i is the collision operator, and the left-hand side is the advection of populations in discrete space and discrete time. Each evolution cycle consists of one advection and one relaxation, and corresponds to one integration time step Δt of the LB method.

There are a number of ways of implementing a suitable collision operator. One approach is to multiply the vector of the old populations F_i by a suitable collision matrix in order to produce the vector of the new populations (Gunstensen&Rothman91 [22], Vergassola [55], Higuera [25]). A simpler approach is to apply a relaxation to each population F_i with a time constant τ (the BGK operator of equation 4.1),

$$(\text{relaxed } F_i) \ = \ F_i \ - \ \frac{(F_i - F_i^{\text{eq}})}{\tau} \tag{4.5}$$

The evolution equation becomes as follows,

$$F_i(\vec{x} + \vec{e}_i\,\Delta t,\, t + \Delta t) \;=\; F_i(\vec{x}, t) \;-\; \frac{(F_i - F_i^{\mathrm{eq}})}{\tau} \tag{4.6}$$

The BGK relaxation is the simplest collision operator that can produce Navier Stokes in the subsonic limit (the ratio of the flow speed divided by the speed of sound must be small). To conserve mass and momentum, the equilibrium populations F_i^{eq} must be chosen so that

$$\begin{aligned} \sum F_i^{\mathrm{eq}} &= \sum F_i \\ \sum F_i^{\mathrm{eq}}\,\vec{e}_i &= \sum F_i\,\vec{e}_i \end{aligned} \tag{4.7}$$

A few additional requirements on the equilibrium populations F_i^{eq} are described in the next section. These additional requirements together with mass and momentum conservation are sufficient to make the lattice Boltzmann method approximate the Navier Stokes equations.

It should be noted that the mapping from the populations F_i to the fluid variables ρ, V_x, V_y is simple (equation 4.2). However, the inverse mapping from the fluid variables ρ, V_x, V_y to the populations F_i is not as simple. The inverse mapping is not needed for the basic LB algorithm, but is useful for implementing initial and boundary conditions as explained in section 4.2.

4.1.1 Hexagonal 7-speed model (d2q7)

The hexagonal 7-speed lattice Boltzmann method is described in detail here. It is denoted "d2q7" following the naming convention of Qian [41]. We consider a hexagonal lattice (see figure 4-2) with six moving populations denoted by $F_i \quad i = 1, \ldots, 6$ and one rest-particle population denoted by F_0. The non-moving population F_0 stays fixed at each node and undergoes only relaxation (collision) at every step. At startup, the populations F_i are initialized from the fluid variables ρ, V_x, V_y (see section 4.2). After initialization, successive steps of relaxation and advection are performed to calculate the F_i and the fluid variables ρ, V_x, V_y at later times. The

relaxation and advection steps are described by the following formulas,

$$\begin{aligned} F_i(\vec{x} + \vec{e}_i\,\Delta t,\, t + \Delta t) &= F_i(\vec{x}, t) + (-1/\tau)\,[F_i(\vec{x}, t) - F_i{}^{\rm eq}(\vec{x}, t)] \\ F_0(\vec{x},\, t + \Delta t) &= F_0(\vec{x}, t) + (-1/\tau)\,[F_0(\vec{x}, t) - F_0{}^{\rm eq}(\vec{x}, t)] \\ i &= 1, \ldots, 6 \end{aligned} \tag{4.8}$$

$$\tau = \frac{1}{2} + \frac{4\Delta t\ \nu}{\Delta x^2} \quad .$$

The relaxation parameter τ is chosen to achieve the desired kinematic viscosity ν given the space and time discretization parameters $\Delta x, \Delta t$. The vector $\vec{e}_i$ stands for the six velocity directions of the hexagonal lattice,

$$\vec{e}_i = \frac{\Delta x}{\Delta t}\,\left(\cos\frac{2\pi\,(i-1)}{6},\ \sin\frac{2\pi\,(i-1)}{6}\right) \tag{4.9}$$

The velocity $\vec{V}(\vec{x}, t)$ and density $\rho(\vec{x}, t)$ are computed from the populations $F_i(\vec{x}, t)$ using the relations,

$$\begin{aligned} \rho(\vec{x}, t) &= \textstyle\sum_{i=0}^{6} F_i(\vec{x}, t) \\ \rho(\vec{x}, t)\,\vec{V}(\vec{x}, t) &= \textstyle\sum_{i=1}^{6} F_i(\vec{x}, t)\,\vec{e}_i \end{aligned} \tag{4.10}$$

The variations of density around its mean value (spatial mean which is constant in time) provide an estimate of the fluid pressure $P(\vec{x}, t)$, according to the following equation,

$$P(\vec{x}, t) = c_s^2\ (\rho(\vec{x}, t) - <\rho>\ ) \tag{4.11}$$

The speed of sound is,

$$c_s = \sqrt{3\,w_0}\ (\Delta x/\Delta t) \tag{4.12}$$

where the coefficient w_0 is discussed below. The equilibrium populations $F_i{}^{\rm eq}(x, t)$ are given by the following equations,

$$\begin{aligned} F_i{}^{\rm eq}(\vec{x}, t) &= \rho(\vec{x}, t)\,\left[w_0 + w_1(\vec{e}_i \cdot \vec{V}) + w_{20}(\vec{e}_i \cdot \vec{V})(\vec{e}_i \cdot \vec{V}) + w_{21}(\vec{V} \cdot \vec{V})\right] \\ F_0{}^{\rm eq}(\vec{x}, t) &= \rho(\vec{x}, t)\,\left[z_0 + z_{21}(\vec{V} \cdot \vec{V})\right] \\ & 6\,w_0 + z_0 = 1 \quad , \\ & w_1 = 1/(3\,c^2) \quad , \quad w_{20} = 2/(3\,c^4) \quad , \quad w_{21} = -1/(6\,c^2) \\ & z_{21} = -1/c^2 \quad , \quad c = \Delta x/\Delta t \end{aligned} \tag{4.13}$$

The above coefficients are chosen so that the Chapman-Enskog expansion of the evolution equation 4.8 matches the Navier Stokes equations (section 4.1.2). In particular, the coefficient w_1 is determined from momentum conservation, the coefficient w_{20} is determined from Galilean invariance (ie. the advection term $(V_x\,\partial V_x/\partial x + V_y\,\partial V_x/\partial y)$ must appear in the Chapman-Enskog expansion with a constant factor equal to one), the coefficient w_{21} is chosen to eliminate the $(\vec{V}\cdot\vec{V})$ dependence of the pressure, and the coefficient z_{21} is chosen to eliminate the $(\vec{V}\cdot\vec{V})$ term in the mass conservation equation. There is some freedom in choosing the remaining coefficients w_0 and z_0, but they must satisfy $6\,w_0 + z_0 = 1$ to conserve mass, and they must be positive for stability purposes. A simple choice is $w_0 = z_0 = (1/7)$.

The computational cycle of the lattice Boltzmann method is organized as follows: The current lattice populations $F_i(\vec{x},t)$ are used to calculate the velocity field $\vec{V}(\vec{x},t)$ and density field $\rho(\vec{x},t)$ according to equation 4.10. These fields are the numerical solution at time t, and they are also used to compute the equilibrium populations $F_i^{\rm eq}(\vec{x},t)$ which are needed to advance the solution. The equilibrium populations $F_i^{\rm eq}(\vec{x},t)$ are used to relax the $F_i(\vec{x},t)$ into "relaxed" populations which are then advected according to equation 4.8 to produce the lattice populations at the next time step. Then the cycle repeats.

4.1.2 Chapman-Enskog expansion

The Chapman-Enskog expansion is outlined here. The goal of the Chapman-Enskog expansion is to derive a set of partial differential equations in terms of ρ and $\rho\vec{V}$ that describe the behavior of the lattice Boltzmann fluid in the limit of $\Delta x, \Delta t$ going to zero. During the Chapman-Enskog expansion, it is assumed that the ratio $\Delta x/\Delta t = c$ is constant, and that the ratio (V/c) is small where V is the macroscopic speed of the fluid. The final result of the Chapman-Enskog expansion is the mass continuity equation and the Navier Stokes momentum equations.

The first step is to Taylor-expand the population variable $F_i(\vec{x}+\vec{e}_i\Delta t,\, t+\Delta t)$ in

the evolution equation 4.8 around the point $(\vec{x}, t)$. This produces an equation whose left-hand side is a Taylor series and whose right-hand side is equal to $(-1/\tau)(F_i - F_i^{\rm eq})$. This equation has the following form,

$$\Delta t \left(\frac{\partial}{\partial t} + \vec{e}_i \cdot \nabla\right) F_i + \frac{\Delta t^2}{2} \left(\frac{\partial}{\partial t} + \vec{e}_i \cdot \nabla\right)^2 F_i + \ldots = \frac{(F_i - F_i^{\rm eq})}{-\tau} \qquad (4.14)$$

The second step is to combine the Taylor series equation 4.14 with the mass and momentum conservation relations (equation 4.10). This produces three equations whose left-hand sides are Taylor series and the right-hand sides vanish because the equilibrium populations $F_i^{\rm eq}$ are chosen to satisfy mass and momentum conservation (for example $\sum_0^6 F_i = \sum_0^6 F_i^{\rm eq}$). The three Taylor series that are derived in this way contain partial derivatives of quantities that are sums and tensors of the populations F_i. The equations have the following form to first order,

$$\partial(\sum_0^6 F_i)/\partial t + \vec{\nabla} \cdot (\sum_1^6 \vec{e}_i F_i) + \ldots = 0 \qquad (4.15)$$

$$\partial(\sum_1^6 \vec{e}_i F_i)/\partial t + \vec{\nabla} \cdot (\sum_1^6 \vec{e}_i \vec{e}_i F_i) + \ldots = 0 \qquad (4.16)$$

If the mass equation is truncated to first-order terms in the derivatives, the resulting equation contains only sums of F_i and no tensors. The sums of F_i can be converted easily to ρ and $\rho\vec{V}$, and this produces the mass continuity equation. The momentum equation must be truncated to second-order terms in the derivatives to produce the Navier Stokes equations. This is necessary because second-order spatial derivatives contribute to the viscosity of the fluid.

A complication arises with the pressure tensor $(\sum \vec{e}_i \vec{e}_i F_i)$ which appears in the momentum equation 4.16. The pressure tensor can not be expressed in terms of ρ and $\rho\vec{V}$ without introducing an approximation of the F_i in terms of ρ and $\rho\vec{V}$. This approximation is necessary in the mass equation also if we include high-order terms in the mass equation.

The Chapman-Enskog expansion approximates the populations $F_i(\vec{x}, t)$ with the equilibrium populations $F_i^{\rm eq}(\vec{x}, t)$ to zero order. Then, a correction is added to first

order,

$$F_i(\vec{x},t) \;=\; F_i^{\rm eq}(\vec{x},t) + F_i^{(1)}(\vec{x},t) \tag{4.17}$$

and so on. The approximation of the F_i can be viewed as another series expansion that is used in parallel with the Taylor series expansion. To retrieve the Navier Stokes equations, it is sufficient to calculate up to first order $F_i^{\rm eq} + F_i^{(1)}$ while keeping up to second-order terms in the Taylor series, as stated previously, in order to retrieve all the viscosity terms.

The correction term $F_i^{(1)}$ is computed from $F_i^{\rm eq}$ using the evolution equation 4.8 Taylor-expanded to first-order with the F_i replaced by the zero-order estimate $F_i^{\rm eq}$ as follows,

$$F_i^{(1)} = -\tau\,\Delta t\,\left[\frac{\partial F_i^{\rm eq}}{\partial t} + \vec{e}_i\cdot\vec{\nabla}F_i^{\rm eq}\right] \tag{4.18}$$

The accuracy of the F_i approximation improves as (V/c) becomes smaller. The above $F_i^{(1)}$ can be used to replace F_i with $F_i^{\rm eq} + F_i^{(1)}$ in the momentum equation 4.16. Further, we express the $F_i^{\rm eq}$ in terms of ρ and $\rho\vec{V}$ in order to derive two partial differential equations in terms of ρ and $\rho\vec{V}$ corresponding to momentum conservation. By choosing the formulas of the equilibrium populations $F_i^{\rm eq}$ appropriately, we can make the momentum equations match the Navier Stokes equations. For example, the equilibrium populations of equation 4.13 produce the following x-momentum equation (to second-order terms),

$$\frac{\partial(\rho V_x)}{\partial t} + \frac{\partial(\rho V_x V_x)}{\partial x} + \frac{\partial(\rho V_x V_y)}{\partial y} = -\frac{\partial(3c^2 w_0\rho)}{\partial x} + \nu\,\nabla^2(\rho V_x) + \mu\,\frac{\partial(\vec{\nabla}\cdot(\rho\vec{V}))}{\partial x} \tag{4.19}$$

$$\nu = \frac{c^2\,\Delta t}{8}\,(2\tau - 1) \qquad \mu = 2\,z_0\,\nu$$

The above viscosity terms differ slightly from the form presented in section 2.4, where the density appears outside the spatial derivatives, for example,

$$\nu\,\rho\,\nabla^2 V_x \qquad \text{and} \qquad \mu\,\rho\,\frac{\partial(\vec{\nabla}\cdot\vec{V})}{\partial x} \tag{4.20}$$

This is not an issue in subsonic flow because the terms $\nu V_x\nabla^2\rho$ (high-order derivatives of density ρ) are very small compared to the other terms.

4.1.3 Stability and accuracy

Formulas that describe the numerical error of the LB method can be obtained in principle by continuing the Chapman-Enskog expansion outlined above. In particular, most of the terms that differ from the Navier Stokes equations in the $F_i^{\rm eq} + F_i^{(1)}$ expansion are multiplied by Δx^2 or Δt^2, which suggests second-order accuracy. However, the terms from the next-order correction $F_i^{(2)}$ must also be considered. Moreover, one must also investigate whether the truncated Chapman-Enskog expansion is adequate to estimate the leading-order error term. This fact is not obvious because the Chapman-Enskog expansion *is not simply a Taylor series expansion*, but a "double" expansion that involves both a Taylor series and another functional series expansion described above. A detailed analysis has not been performed yet.

Leaving aside the theoretical difficulties, experimental evidence presented in sections 4.4 and 4.5, shows that the LB method is second-order accurate in space and in time. In the future, it would be very interesting to calculate theoretically the constants of the leading-order error terms for the different LB models (the d2q7 above, and the d2q9 and d2q15 described later), and to test whether the theoretical error constants agree with the experimental results.

Stability conditions for the LB method are not known in general. A few necessary conditions are as follows. First, a CFL condition for explicit methods requires that the ratio of the flow speed divided by the numerical speed $V/(\Delta x/\Delta t)$ should be less than one. [2] In addition, a subsonic flow condition must be satisfied that the ratio of the flow speed divided by the speed of sound V/c_s should be less than one. It should be noted that the CFL condition applies generally to all explicit methods (see section 3.3.1), but the subsonic flow condition is an additional requirement of the present lattice Boltzmann approach.

[2] The CFL condition also requires that the ratio of the sound speed divided by the numerical speed $c_s/(\Delta x/\Delta t)$ should be less than one. This is always true in the case of lattice Boltzmann because $c_s = \sqrt{3\,w_0}(\Delta x/\Delta t)$ and because of the constrains on the density coefficients w_0, z_0 (for example, see equation 4.22).

Another stability condition is that the density coefficients w_0, z_0, y_0 of the equilibrium population formulas must be positive. [3] This fact can be proven by considering the norm of the vector of populations F_i, and by requiring that the norm does not grow after the relaxation (collision operator) is applied. However, the algebra is rather complicated and is omitted here. It is very easy to verify experimentally that non-positive density coefficients w_0, z_0, y_0 lead to instabilities.

The requirement for positivity of the density coefficients w_0, z_0, y_0 can be combined with other formulas to deduce further conditions. For example, in the case of the 9-speed d2q9 model, the mass conservation formula is

$$4w_0 + 4y_0 + z_0 = 1 \tag{4.21}$$

Using $y_0 = w_0/4$ gives,

$$w_0 \leq \frac{1}{5} \tag{4.22}$$

as an upper bound on the coefficient w_0. Actually, a more stringent bound can be obtained by considering the formula for the bulk viscosity,

$$\mu = 2\nu(1 - 3w_0 - 6y_0) \tag{4.23}$$

The second law of thermodynamics applied to the dissipation of energy during the compression of fluid elements (Landau&Lifshitz [32, p.45]) requires that

$$\mu \geq \frac{\nu}{3} \tag{4.24}$$

which gives

$$w_0 + 2y_0 \leq \frac{5}{18} \tag{4.25}$$

or using the choice $y_0 = w_0/4$,

$$w_0 \leq \frac{5}{27} \tag{4.26}$$

The above formulas are necessary conditions for the stability of the lattice Boltzmann method.

[3] The density coefficient y_0 is used in the orthogonal d2q9 model of section 4.3. The density coefficient y_0 of the d2q9 model should be preferably chosen $y_0 = w_0/4$ following the ratio of the other coefficients such as $y_1 = w_1/4$.

4.2 Initial and boundary conditions

Having reviewed the basic theory of the LB method, it is now appropriate to discuss how to implement accurate initial and boundary conditions for the LB method. The basic idea is to find a good way of calculating the populations F_i from the fluid variables ρ, V_x, V_y.

My approach is to combine the standard collision operator of the lattice Boltzmann method with a new extended collision operator. This combination is referred to as the hybrid method, and is described below. An alternative approach is to truncate the Chapman-Enskog expansion. In theory, the infinite series of the Chapman-Enskog expansion produces exactly the inverse mapping; however, in practice the Chapman-Enskog expansion must be truncated. Furthermore, the obvious truncation of the Chapman-Enskog expansion does not perform very well (numerical tests of the zero-th and the first-order truncated series are given in section 4.4). However, if the first-order truncated series is modified appropriately, it produces an expression which is identical to the hybrid method. This equivalence of the hybrid method and a modified Chapman-Enskog expansion was first noticed by Dominique d'Humières who kindly communicated this result to the author.

4.2.1 Previous approaches and related work

Before presenting the hybrid method and the extended collision operator, it is useful to review how initial and boundary conditions for the LB method have been traditionally implemented.

Traditionally, the use of an accurate inverse mapping for the lattice Boltzmann populations has been avoided both for initial value and for boundary value problems. In the case of initial value problems, when the fluid density and velocity ρ, V_x, V_y are specified at time zero and the goal is to calculate ρ, V_x, V_y at later times, the populations F_i can be initialized equal to the equilibrium values $F_i{}^{\mathrm{eq}}$ which are known

in terms of ρ, V_x, V_y. The error that results from this approximation can be overcome by discarding the first few steps and measuring the parameters of the flow afterwards (recalibrating the solution). This is often done in the literature without further discussion. The problem with recalibration is that a slightly different problem is solved than the original ρ, V_x, V_y. By contrast, traditional methods such as finite differences do not need any recalibration. Thus, to put the lattice Boltzmann method on equal footing with other methods (for numerical testing in particular) it is desirable to have an accurate means of calculating the populations F_i from the initial values of ρ, V_x, V_y.

In the case of boundary conditions, there are techniques that avoid the inverse mapping as in the case of initial conditions. In particular, the velocity of the fluid can be forced to zero at non-slip wall boundaries by imposing a non-slip bounce-back of the populations F_i. However, the location of the wall is not always well defined (see Cornubert&et al. [12], Ginzbourg&Adler [21] for a discussion of the actual location of the wall as a function of the simulation parameters for some simple flows). In the case of boundary conditions with non-zero velocity, such as the driven cavity problem Peyret&Taylor [38, p.199], the velocity at the boundary can be controlled by inserting momentum (forcing) in every step as is done in lattice gas automata. This type of forcing is somewhat ad-hoc however, and is often inaccurate, and requires recalibration of the simulation parameters. In the case of an arbitrary velocity specification at the boundary, such as the fluid flows of section 4.5, the forcing techniques and the recalibration become very difficult. Thus, it is desirable to have an accurate means of calculating the populations F_i at a boundary node from the fluid variables ρ, V_x, V_y that are specified at this node.

4.2.2 Hybrid method and extended collision operator

The calculation of the populations F_i from fluid variables ρ, V_x, V_y is now described. For this purpose, an extended collision operator is introduced (denoted d2q7X) which

differs from the standard collision operator in the equilibrium population formulas. The evolution equation remains as before, and can be written as follows,

$$F_i(\vec{x} + \vec{e}_i\,\Delta t,\, t + \Delta t) \;=\; F_i(\vec{x}, t) + \frac{F_i(\vec{x}, t) - F_i^{*\mathrm{eq}}(\vec{x}, t)}{-\tau^*} \tag{4.27}$$

The relaxation parameter of the extended collision operator is denoted τ^* to distinguish it from the relaxation parameter τ of the standard collision operator (and accordingly for other parameters shown below).

The important idea is that the equilibrium population formulas $F_i^{*\mathrm{eq}}$ of the extended collision operator include additional terms (shown below) so that the viscosity can be controlled independently from the relaxation parameter τ^*. Thus, τ^* can be set equal to one, which implies that the F_i are replaced by the $F_i^{*\mathrm{eq}}$ at each step. In other words, the old F_i are not needed anymore, and the $F_i^{*\mathrm{eq}}$ provide a direct mapping from the flow variables ρ, V_x, V_y to the new populations F_i.

The extended collision operator is used everywhere (all the fluid nodes) at startup, but only at the boundary nodes during the simulation. After the first step, the standard collision operator is used at the inner (non-boundary) nodes. This combination of the two operators is referred to as the hybrid method here (denoted d2q7H in the case of the hexagonal model). It is valid to combine two different collision operators as long as the two operators have the same transport coefficients (shear and bulk viscosity) which is true here.

The equilibrium population formulas $F_i^{*\mathrm{eq}}$ of the extended collision operator include terms which are based on the gradients of the fluid velocity, and are motivated by equation 2.5.1 of Wolfram [58]. The equilibrium population formulas $F_i^{*\mathrm{eq}}$ are as follows,

$$\begin{aligned}
F_i^{*\mathrm{eq}}(\vec{x}, t) \;&=\; \rho(\vec{x}, t)\,\Big[w_0 + w_1(\vec{e}_i \cdot \vec{V}) + w_{20}(\vec{e}_i \cdot \vec{V})(\vec{e}_i \cdot \vec{V}) + w_{21}(\vec{V} \cdot \vec{V})\Big] + \\
&\quad w_{31}\,(\vec{e}_i \cdot \vec{\nabla}(\vec{e}_i \cdot \rho\vec{V})) \;+\; w_{32}\,(\vec{\nabla} \cdot \rho\vec{V}) \quad , \\
&\quad i = 1, \ldots, 6 \\
F_0^{*\mathrm{eq}}(\vec{x}, t) \;&=\; \rho(\vec{x}, t)\,\Big[z_0 + z_{21}(\vec{V} \cdot \vec{V})\Big] + z_{32}\,(\vec{\nabla} \cdot \rho\vec{V})
\end{aligned} \tag{4.28}$$

$$3\,c^2\,w_{31} + 6\,w_{32} + z_{32} \;=\; 0$$

The velocity gradients in the above equation (the terms with coefficients w_{31}, w_{32}, z_{32}) are computed using finite differences unless they are known by other means; for example some of the velocity gradients may be known at the boundary nodes (see section 4.5). The coefficients $w_0, w_1, w_{20}, w_{21}, z_0, z_{21}$ have the same values as in the standard collision operator d2q7 (equation 4.13). It is worth noting that the velocity gradient terms of equation 4.28 can be **viewed as a correction** to the equilibrium population formulas,

$$F_i^{*\mathrm{eq}} = F_i^{\mathrm{eq}} + F_i^{(1\mathrm{X})} \tag{4.29}$$

where

$$F_i^{(1\mathrm{X})} = w_{31}\,(\vec{e}_i \cdot \vec{\nabla}(\vec{e}_i \cdot \rho\vec{V})) \;+\; w_{32}\,(\vec{\nabla} \cdot \rho\vec{V}) \tag{4.30}$$

The above formula is used in the next section to relate the extended collision operator to a truncated Chapman-Enskog expansion.

Using the Chapman-Enskog expansion, the shear and bulk viscosities of the extended collision operator can be calculated,

$$\nu^* = \frac{c^2\,\Delta t}{8}\,(2\tau^* - 1) \;-\; \frac{3\,c^4\,w_{31}}{4} \tag{4.31}$$

$$\mu^* = \frac{c^2\,\Delta t}{4}\,(2\tau^* - 1)\,z_0 \;-\; \frac{3\,c^4\,w_{31}}{2} \;-\; 3\,c^2\,w_{32}$$

When τ^* is set equal to one, the coefficient w_{31} is chosen to achieve the desired shear viscosity given the discretization parameters $\Delta x, \Delta t$. The coefficient w_{32} is chosen to achieve the desired bulk viscosity, and the coefficient z_{32} is chosen to enforce the relation $(3\,c^2\,w_{31} + 6\,w_{32} + z_{32}) = 0$ which corresponds to mass conservation.

In the case of the hybrid method (when the standard and extended collision operators are used in the same computation), the bulk viscosity of equation 4.31 is chosen equal to the bulk viscosity of the standard collision operator given by equation 4.19 (similarly for the shear viscosity). Also, the relaxation parameter τ^* is set equal to

1.0. In this case, the coefficients w_{31}, w_{32}, z_{32} simplify as follows,

$$w_{31} = \frac{(1-\tau)\Delta t}{3c^2} \tag{4.32}$$

$$w_{32} = -w_0(1-\tau)\Delta t$$

$$z_{32} = -z_0(1-\tau)\Delta t$$

It should be noted that the extended collision operator is accurate when used for initial and boundary conditions, but it is not accurate when iterated many times. It appears that the finite differences which are used by the extended collision operator produce an error in viscosity which means that the computed solution decays at a slightly different rate than desired. The error accumulates with successive iterations, and the method does not approximate the solution as Δt goes to zero (see figure 4-6 in section 4.4.2). However, this is not a problem in practice because the extended collision operator is only used at startup and subsequently only at the boundary nodes.

Finally, another issue worth mentioning is the initialization of the density at startup. Quite often, the pressure $P(x,y)$ is specified at startup. Then, the density $\rho(x,y)$ must be computed from the pressure,

$$\rho(x,y) \;\; = \;\; <\rho> \;\; + (\frac{1}{c_s^2})\, P(x,y) \tag{4.33}$$

where c_s is the speed of sound, $<\rho>$ is the constant average density, and P is the pressure (with the constant average pressure subtracted so that $<P> \;\; = \;\; 0$). It is very important **not** to initialize the density to be constant. The density must follow the initial pressure gradients according to equation 4.33; otherwise large density waves and error transients may result. Once the density and velocity ρ, V_x, V_y are specified correctly, the populations F_i can be calculated from ρ, V_x, V_y using the extended collision operator described above.

4.2.3 Truncated Chapman-Enskog expansion

An alternative way of deriving the hybrid method is to employ a truncated Chapman-Enskog expansion and to perform additional manipulations. Below, the zero-order and first-order truncated Chapman-Enskog expansions are described, and then it is shown how to modify and simplify the first-order expansion in order to obtain the hybrid method.

The zero-order expansion, denoted by d2q7F0, approximates the populations F_i with the equilibrium value F_i^{eq}. As stated earlier, this approximation is used very often in the literature, and it is accompanied by recalibration of the solution after the first few steps are discarded (initial transients). The zero-order expansion is tested experimentally in section 4.4; however, recalibration is not performed there because the goal is to compare the accuracy of calculating the populations F_i from the fluid variables ρ, V_x, V_y.

The first-order expansion, denoted by d2q7F1, approximates the populations F_i with the Chapman-Enskog expansion truncated to first-order,

$$\begin{aligned} F_i &= F_i^{\text{eq}} + F_i^{(1)} \\ F_i^{(1)} &= -\tau\,\Delta t\left[\frac{\partial F_i^{\text{eq}}}{\partial t} + \vec{e}_i \cdot \vec{\nabla} F_i^{\text{eq}}\right] \end{aligned} \tag{4.34}$$

A differentiation of the equilibrium population formulas (equation 4.13) provides formulas for the derivatives of F_i^{eq} in terms of the derivatives of the fluid variables ρ, V_x, V_y. The derivatives of ρ, V_x, V_y are known in some cases (for example in exactly solvable fluid flow problems), but in general the derivatives must be estimated using finite differences. The initialization tests of section 4.4 employ finite differences. In particular, the time derivatives of ρ, V_x, V_y are estimated using the Navier Stokes momentum and continuity equations, and the spatial derivatives of ρ, V_x, V_y are estimated using spatial finite differences. I have also tested the different initialization methods using the exact values of the derivatives, and the results are qualitatively the same as those reported in section 4.4.

In section 4.4, it is shown that both d2q7F0 and d2q7F1 produce significant errors in initialization. It is a little surprising that the first-order Chapman-Enskog correction does not perform well, but there is an easy explanation. We observe that the correction term $F_i^{(1)}$ of equation 4.34 does not conserve momentum. This means that the velocity field that results from equation 4.34 is different from the original velocity field. The conservation relations that correspond to equation 4.34 are as follows,

$$\sum_i F_i^{(1)} = (\vec{\nabla} \cdot \rho\vec{V}) + \frac{\partial \rho}{\partial t}$$

$$\sum_i F_i^{(1)} e_{ix} = \frac{\partial(\rho V_x)}{\partial t} + c_s^2 \frac{\partial \rho}{\partial x} + \frac{\partial(\rho V_x V_x)}{\partial x} + \frac{\partial(\rho V_x V_y)}{\partial y} \tag{4.35}$$

and a similar equation for $\sum_i F_i^{(1)} e_{iy}$. Therefore, mass is conserved via the macroscopic continuity equation, but momentum is not conserved. On the other hand, the above equations suggest an easy way to fix the problem: We simply add a viscosity Laplacian term so that momentum will be conserved via the Navier Stokes momentum equation. The new (modified) Chapman-Enskog correction term, denoted by $F_i^{(1\mathrm{M})}$, is as follows [4],

$$F_i^{(1\mathrm{M})} = -\tau\, \Delta t \left[\frac{\partial F_i^{\mathrm{eq}}}{\partial t} + \vec{e}_i \cdot \vec{\nabla} F_i^{\mathrm{eq}} + (-\nu/(3c^2))\, \nabla^2 (\vec{e}_i \cdot \vec{V}) \right] \tag{4.36}$$

In the numerical tests of section 4.4, the above equation is referred to as d2q7F1M. The numerical tests show that d2q7F1M is very accurate for initialization purposes. In practice however, the d2q7F1M method is rather cumbersome to apply because it requires the calculation of many derivatives, including a time derivative and a Laplacian term.

Fortunately, equation 4.36 can be simplified greatly by neglecting second-order terms in the Mach number. This means that only terms up to first order in (V/c) are kept in the Chapman-Enskog expansion, and terms proportional to $(V/c)^2$ are

[4] The addition of a viscosity Laplacian term to the first-order Chapman-Enskog expansion (for the purpose of conserving momentum) does not change the derivation of the Navier Stokes equations via the Chapman-Enskog procedure because the corresponding corrections are higher-order derivatives than the Navier Stokes equations.

discarded because they are small. In addition, the time derivatives are replaced by space derivatives using the macroscopic mass and momentum equations. Examples of this kind of expansion can be found in Frisch [20] and d'Humières [15]. Thus, equation 4.36 simplifies to,

$$F_i^{(1S)} = -\tau\,\Delta t\,\left[\frac{1}{3c^2}\vec{e}_i\cdot\vec{\nabla}(\vec{e}_i\cdot\rho\vec{V}) - w_0(\vec{\nabla}\cdot\rho\vec{V})\right] \tag{4.37}$$

and similarly for the rest particle population,

$$F_0^{(1S)} = -\tau\,\Delta t\,\left[-z_0(\vec{\nabla}\cdot\rho\vec{V})\right] \tag{4.38}$$

In section 4.4, it is shown that the simplified equation 4.37 is as accurate as the original equation 4.36 for initialization purposes.

The above formulas look suspiciously similar to the hybrid method that was described in the last section. In fact, it is easy to verify that equations 4.37 and 4.38 produce identical results with the hybrid method. If equation 4.37 is used to initialize the populations F_i as $F_i = F_i^{\rm eq} + F_i^{(1S)}$, and the first relaxation step is performed, then the resulting populations which are advected (denoted $\bar{F}_i$) are as follows,

$$\bar{F}_i = (F_i^{\rm eq} + F_i^{(1S)}) + (-1/\tau)F_i^{(1S)} \tag{4.39}$$

$$\bar{F}_i = F_i^{\rm eq} + (1-\tau)\Delta t\left[\frac{1}{3c^2}(\vec{e}_i\cdot\vec{\nabla}(\vec{e}_i\cdot\rho\vec{V})) - w_0(\vec{\nabla}\cdot\rho\vec{V})\right] \tag{4.40}$$

The above populations are identical to the populations that are advected after a relaxation step using the extended collision operator (equation 4.29) when the simplified values of w_{31}, w_{32}, z_{32} for the hybrid method are used (equation 4.32). This shows that the simplified truncated first-order Chapman-Enskog expansion is equivalent to the extended collision operator.

In the next section, LB models are described which are appropriate for orthogonal grids in two and in three dimensions. The results of this section are applied straightforwardly to the orthogonal LB models.

4.3 Lattice Boltzmann for orthogonal grids

The ideas discussed in the previous sections using the hexagonal 7-speed model can be applied straightforwardly to other lattice Boltzmann models. Here, the orthogonal 9-speed model in two dimensions is described.

4.3.1 Two-dimensional 9-speed model (d2q9)

The orthogonal 9-speed model is abbreviated by the symbol d2q9 following the convention of Qian [41]. An orthogonal lattice (see figure 4-1) with nine populations at each node is used. The population F_0 is non-moving, the populations F_i^{II} $i = 2, 4, 6, 8$ move along the diagonal directions at the speed $\sqrt{2}c$, and the populations F_i^{I} $i = 1, 3, 5, 7$ move along the vertical and horizontal directions at the speed $c = \Delta x/\Delta t$. The relaxation and advection steps are given by the following formulas,

$$\begin{aligned} F_i(\vec{x} + \vec{e}_i\,\Delta t,\, t + \Delta t) &= F_i(\vec{x}, t) + (-1/\tau)\,[F_i(\vec{x}, t) - F_i{}^{\mathrm{eq}}(\vec{x}, t)] \\ F_0(\vec{x},\, t + \Delta t) &= F_0(\vec{x}, t) + (-1/\tau)\,[F_0(\vec{x}, t) - F_0{}^{\mathrm{eq}}(\vec{x}, t)] \\ i &= 1, \ldots, 8 \end{aligned} \tag{4.41}$$

$$\tau = \frac{1}{2} + \frac{3\Delta t\ \nu}{\Delta x^2}$$

The relaxation parameter τ is chosen to achieve the desired kinematic viscosity ν given the space and time discretization parameters $\Delta x, \Delta t$. The vector $\vec{e}_i$ stands for the eight velocity directions of the orthogonal (square) lattice,

$$\vec{e}_i = \frac{\Delta x}{\Delta t}\left(\cos\frac{2\pi(i-1)}{8},\ \sin\frac{2\pi(i-1)}{8}\right) \quad . \tag{4.42}$$

The velocity $\vec{V}(\vec{x}, t)$ and density $\rho(\vec{x}, t)$ are computed from the populations $F_i(\vec{x}, t)$ using the relations,

$$\begin{aligned} \rho(\vec{x}, t) &= \textstyle\sum_{i=0}^{8} F_i(\vec{x}, t) \\ \rho(\vec{x}, t)\,\vec{V}(\vec{x}, t) &= \textstyle\sum_{i=1}^{8} F_i(\vec{x}, t)\,\vec{e}_i \end{aligned} \tag{4.43}$$

The variations of density around its mean value (spatial mean which is constant in time) provide an estimate of the fluid pressure $P(\vec{x},t)$, according to the following equation,

$$P(\vec{x},t) \;=\; c_s^2\;(\rho(\vec{x},t) - <\rho>)\quad . \tag{4.44}$$

The speed of sound is,

$$c_s \;=\; \sqrt{(2\,w_0 \,+ 4\,y_0)}\,(\Delta x/\Delta t) \tag{4.45}$$

where the coefficients w_0, y_0 are discussed below. The $F_i^{\rm eq}(x,t)$ equilibrium populations are given by the following equations,

$$\begin{aligned} F_i^{\rm eq II} &= \rho\,\left[y_0 + y_1(\vec{e}_i\cdot\vec{V}) + y_{20}(\vec{e}_i\cdot\vec{V})(\vec{e}_i\cdot\vec{V}) + y_{21}(\vec{V}\cdot\vec{V})\right] \\ F_i^{\rm eq I} &= \rho\,\left[w_0 + w_1(\vec{e}_i\cdot\vec{V}) + w_{20}(\vec{e}_i\cdot\vec{V})(\vec{e}_i\cdot\vec{V}) + w_{21}(\vec{V}\cdot\vec{V})\right] \\ F_0{}^{\rm eq} &= \rho\,\left[z_0 + z_{21}(\vec{V}\cdot\vec{V})\right] \end{aligned} \tag{4.46}$$

$$\begin{aligned} &4\,w_0 \,+\, 4\,y_0 \,+\, z_0 = 1\quad , \\ &y_1 = 1/(12\,c^2)\quad ,\quad y_{20} = 1/(8\,c^4)\quad ,\quad y_{21} = -1/(24\,c^2) \\ &w_1 = 1/(3\,c^2)\quad ,\quad w_{20} = 1/(2\,c^4)\quad ,\quad w_{21} = -1/(6\,c^2) \\ &z_{21} = -2/(3\,c^2)\quad ,\quad c = \Delta x/\Delta t \end{aligned}$$

The coefficient y_0 is chosen $y_0 = (1/4)\,w_0$ for simplicity. The coefficient w_0 can be varied to adjust the speed of sound and the bulk viscosity within the stability constraints $w_0 > 0$ and $z_0 > 0$. The shear and bulk viscosity of the d2q9 collision operator have the following values (calculated using the Chapman-Enskog procedure),

$$\nu = \frac{c^2\,\Delta t}{6}\,(2\tau - 1) \tag{4.47}$$

$$\mu = \frac{c^2\,\Delta t}{3}\,(2\tau - 1)\,(1 - 3\,w_0 - 6\,y_0)$$

The extended collision operator for the orthogonal 9-speed model (d2q9X) is derived similarly to the hexagonal model of section 4.2.2. Two additional terms based on gradients of the fluid velocity are included in the equilibrium population formulas. Everything else, including all the coefficients $w_1, y_1, w_{20}, \ldots$ of the standard collision

operator d2q9 remain the same. The equilibrium population formulas for d2q9X are as follows,

$$\begin{aligned} F_i^{*\mathrm{eq}II} &= \rho\,\Big[y_0 + y_1(\vec{e}_i\cdot\vec{V}) + y_{20}(\vec{e}_i\cdot\vec{V})(\vec{e}_i\cdot\vec{V}) + y_{21}(\vec{V}\cdot\vec{V})\Big] + \\ &\quad y_{31}\,(\vec{e}_i\cdot\vec{\nabla}(\vec{e}_i\cdot\rho\vec{V}))\;+\;y_{32}\,(\vec{\nabla}\cdot\rho\vec{V}) \\ F_i^{*\mathrm{eq}I} &= \rho\,\Big[w_0 + w_1(\vec{e}_i\cdot\vec{V}) + w_{20}(\vec{e}_i\cdot\vec{V})(\vec{e}_i\cdot\vec{V}) + w_{21}(\vec{V}\cdot\vec{V})\Big] + \\ &\quad w_{31}\,(\vec{e}_i\cdot\vec{\nabla}(\vec{e}_i\cdot\rho\vec{V}))\;+\;w_{32}\,(\vec{\nabla}\cdot\rho\vec{V}) \\ F_0^{*\mathrm{eq}} &= \rho\,\Big[z_0 + z_{21}(\vec{V}\cdot\vec{V})\Big] + z_{32}\,(\vec{\nabla}\cdot\rho\vec{V}) \end{aligned} \tag{4.48}$$

$$2\,c^2\,w_{31}\;+\;4\,w_{32}\;+\;4\,c^2\,y_{31}\;+\;4\,y_{32}+z_{32}\;=\;0 \tag{4.49}$$

$$y_{31} = w_{31}/4 \tag{4.50}$$

Equation 4.49 is necessary for mass conservation and can be used to determine the coefficient z_{32}. Equation 4.50 is necessary to remove an unwanted (anisotropic) momentum diffusion term in the Chapman-Enskog expansion. The velocity gradients of the extended collision operator must be computed using finite differences unless they are known by other means.

The shear and bulk viscosities of the d2q9X operator have the following values (calculated using the Chapman-Enskog procedure),

$$\nu^* = \frac{c^2\,\Delta t}{6}\,(2\tau^* - 1)\;-\;c^4\,w_{31} \tag{4.51}$$

$$\mu^* = \frac{c^2\,\Delta t}{3}\,(2\tau^* - 1)\,(1 - 3\,w_0 - 6\,y_0)\;-\;2\,c^4\,w_{31}\;-\;2\,c^2(w_{32} + 2\,y_{32})$$

The parameter y_{32} is chosen $y_{32} = w_{32}/4$ for simplicity. Once the relaxation parameter τ^* is set equal to one, the coefficient w_{31} is chosen to achieve the desired kinematic viscosity given the discretization parameters $\Delta x, \Delta t$. The coefficient w_{32} is chosen to achieve the desired bulk viscosity. In the case of the hybrid method d2q9H, the bulk viscosity of equation 4.51 is chosen equal to the bulk viscosity of the standard collision operator given by equation 4.47.

4.3.2 Three-dimensional 15-speed model (d3q15)

The orthogonal 15-speed model is abbreviated by the symbol d3q15 following the convention of Qian [41]. A 3-dimensional cubic lattice with 15 populations at each node is used as shown in figure 4-3. The populations F_i^{II} $i = 7, 8, 9, 10, 11, 12, 13, 14$ move along the diagonal directions at the speed $\sqrt{2}c$, and the populations F_i^{I} $i = 1, 2, 3, 4, 5, 6$ move along the non-diagonal directions at the speed $c = \Delta x/\Delta t$. The non-moving population is F_0. The relaxation and advection steps are given by the following formulas,

$$\begin{aligned} F_i(\vec{x} + \vec{e}_i\,\Delta t,\, t + \Delta t) &= F_i(\vec{x}, t) + (-1/\tau)\,[F_i(\vec{x}, t) - F_i{}^{\rm eq}(\vec{x}, t)] \\ F_0(\vec{x},\, t + \Delta t) &= F_0(\vec{x}, t) + (-1/\tau)\,[F_0(\vec{x}, t) - F_0{}^{\rm eq}(\vec{x}, t)] \\ i &= 1, \ldots, 8 \end{aligned} \tag{4.52}$$

$$\tau = \frac{1}{2} + \frac{3\Delta t\ \nu}{\Delta x^2}$$

The relaxation parameter τ is chosen to achieve the desired kinematic viscosity ν given the space and time discretization parameters $\Delta x, \Delta t$. The vector $\vec{e}_i$ stands for the 14 velocity directions of the 3-dimensional cubic lattice, as shown in figure 4-3.

The velocity $\vec{V}(\vec{x}, t)$ and density $\rho(\vec{x}, t)$ are computed from the populations $F_i(\vec{x}, t)$ using the relations,

$$\begin{aligned} \rho(\vec{x}, t) &= \textstyle\sum_{i=0}^{8} F_i(\vec{x}, t) \\ \rho(\vec{x}, t)\,\vec{V}(\vec{x}, t) &= \textstyle\sum_{i=1}^{8} F_i(\vec{x}, t)\,\vec{e}_i \end{aligned} \tag{4.53}$$

The variations of density around its mean value (spatial mean which is constant in time) provide an estimate of the fluid pressure $P(\vec{x}, t)$, according to the following equation,

$$P(\vec{x}, t) = c_s^2\ (\rho(\vec{x}, t) - <\rho>)\quad . \tag{4.54}$$

The speed of sound is,

$$c_s = \sqrt{(2\,w_0 + 8\,y_0)}\ (\Delta x/\Delta t) \tag{4.55}$$

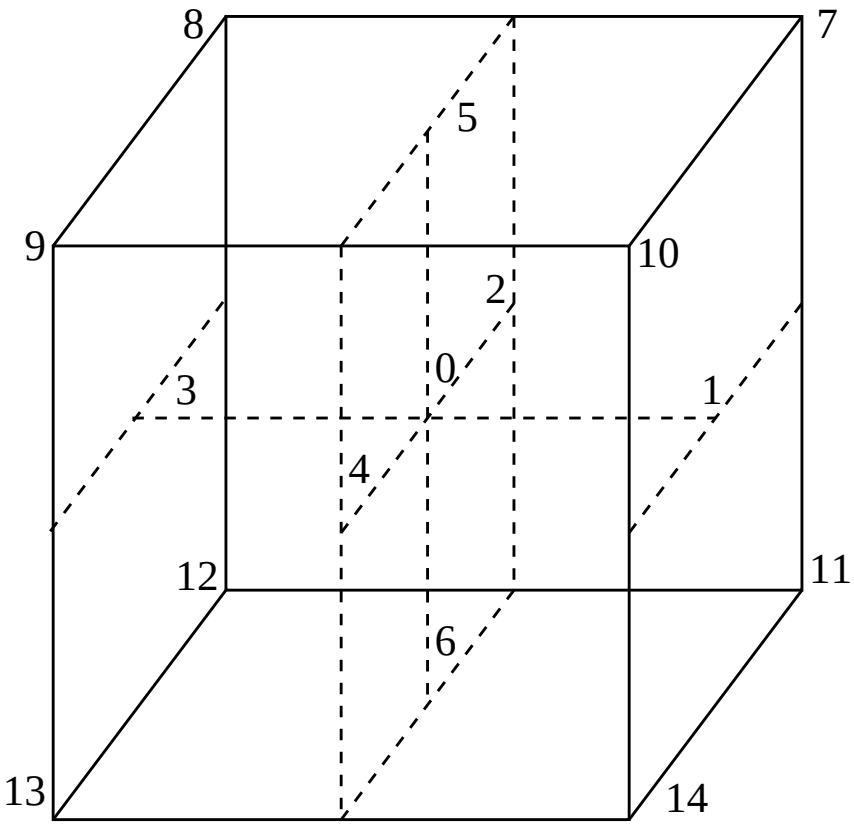

Figure 4-3: Velocity directions for lattice Boltzmann d3q15 in three-dimensions.

where the coefficients w_0, y_0 are discussed below. The equilibrium populations $F_i^{\rm eq}(x,t)$ are given by the following equations,

$$\begin{aligned} F_i^{\rm eq II} &= \rho \left[y_0 + y_1(\vec{e}_i \cdot \vec{V}) + y_{20}(\vec{e}_i \cdot \vec{V})(\vec{e}_i \cdot \vec{V}) + y_{21}(\vec{V} \cdot \vec{V}) \right] \\ F_i^{\rm eq I} &= \rho \left[w_0 + w_1(\vec{e}_i \cdot \vec{V}) + w_{20}(\vec{e}_i \cdot \vec{V})(\vec{e}_i \cdot \vec{V}) + w_{21}(\vec{V} \cdot \vec{V}) \right] \\ F_0{}^{\rm eq} &= \rho \left[z_0 + z_{21}(\vec{V} \cdot \vec{V}) \right] \end{aligned} \tag{4.56}$$

$$\begin{aligned} & 6\,w_0 \,+\, 8\,y_0 \,+\, z_0 = 1 \ , \\ & y_1 = 1/(12\,c^2) \ , \ \ y_{20} = 1/(16\,c^4) \ , \ \ y_{21} = -1/(48\,c^2) \\ & w_1 = 1/(3\,c^2) \ , \ \ w_{20} = 1/(2\,c^4) \ , \ \ w_{21} = -1/(6\,c^2) \\ & z_{21} = -1/(3\,c^2) \ , \ \ c = \Delta x/\Delta t \end{aligned}$$

The coefficient y_0 is chosen $y_0 = (1/8)\,w_0$ for simplicity. The coefficient w_0 can be varied to adjust the speed of sound and the bulk viscosity within the stability constraints $w_0 > 0$ and $z_0 > 0$. The shear and bulk viscosity of the d3q15 collision

operator have the following values (calculated using the Chapman-Enskog procedure),

$$\nu = \frac{c^2\,\Delta t}{6}\,(2\tau - 1) \tag{4.57}$$

$$\mu = \frac{c^2\,\Delta t}{3}\,(2\tau - 1)\,(1 - 3\,w_0 - 12\,y_0)$$

The extended collision operator (d3q15X) for the orthogonal 15-speed model is derived similarly to the hexagonal model of section 4.1.1. Two additional terms based on gradients of the fluid velocity are included in the equilibrium population formulas. Everything else, including all the coefficients $w_1, y_1, w_{20}, \ldots$ of the standard collision operator d3q15 remain the same. The equilibrium population formulas for d3q15X are as follows,

$$\begin{aligned}
F_i^{\mathrm{eq}II} &= \rho\,\left[y_0 + y_1(\vec{e}_i\cdot\vec{V}) + y_{20}(\vec{e}_i\cdot\vec{V})(\vec{e}_i\cdot\vec{V}) + y_{21}(\vec{V}\cdot\vec{V})\right] + \\
&\quad y_{31}\,(\vec{e}_i\cdot\vec{\nabla}(\vec{e}_i\cdot\rho\vec{V})) \;+\; y_{32}\,(\vec{\nabla}\cdot\rho\vec{V}) \\
F_i^{\mathrm{eq}I} &= \rho\,\left[w_0 + w_1(\vec{e}_i\cdot\vec{V}) + w_{20}(\vec{e}_i\cdot\vec{V})(\vec{e}_i\cdot\vec{V}) + w_{21}(\vec{V}\cdot\vec{V})\right] + \\
&\quad w_{31}\,(\vec{e}_i\cdot\vec{\nabla}(\vec{e}_i\cdot\rho\vec{V})) \;+\; w_{32}\,(\vec{\nabla}\cdot\rho\vec{V}) \\
F_0^{\mathrm{eq}} &= \rho\,\left[z_0 + z_{21}(\vec{V}\cdot\vec{V})\right] + z_{32}\,(\vec{\nabla}\cdot\rho\vec{V})
\end{aligned} \tag{4.58}$$

$$2\,c^2\,w_{31} \;+\; 6\,w_{32} \;+\; 8\,c^2\,y_{31} \;+\; 8\,y_{32} + z_{32} \;=\; 0 \tag{4.59}$$

$$y_{31} = w_{31}/8 \tag{4.60}$$

Equation 4.59 is necessary for mass conservation and can be used to determine the coefficient z_{32}. Equation 4.60 is necessary to remove an unwanted (anisotropic) momentum diffusion term in the Chapman-Enskog expansion.

The shear and bulk viscosity of the d3q15X operator have the following values (calculated using the Chapman-Enskog procedure),

$$\nu = \frac{c^2\,\Delta t}{6}\,(2\tau - 1) \;-\; c^4\,w_{31} \tag{4.61}$$

$$\mu = \frac{c^2\,\Delta t}{3}\,(2\tau - 1)\,(1 - 3\,w_0 - 12\,y_0) \;-\; 2\,c^4\,w_{31} \;-\; c^2\,(2\,w_{32} + 8\,y_{32})$$

The coefficient y_{32} is chosen $y_{32} = w_{32}/8$ for simplicity. Once the relaxation parameter τ is set equal to one, the coefficient w_{31} is chosen to achieve the desired kinematic

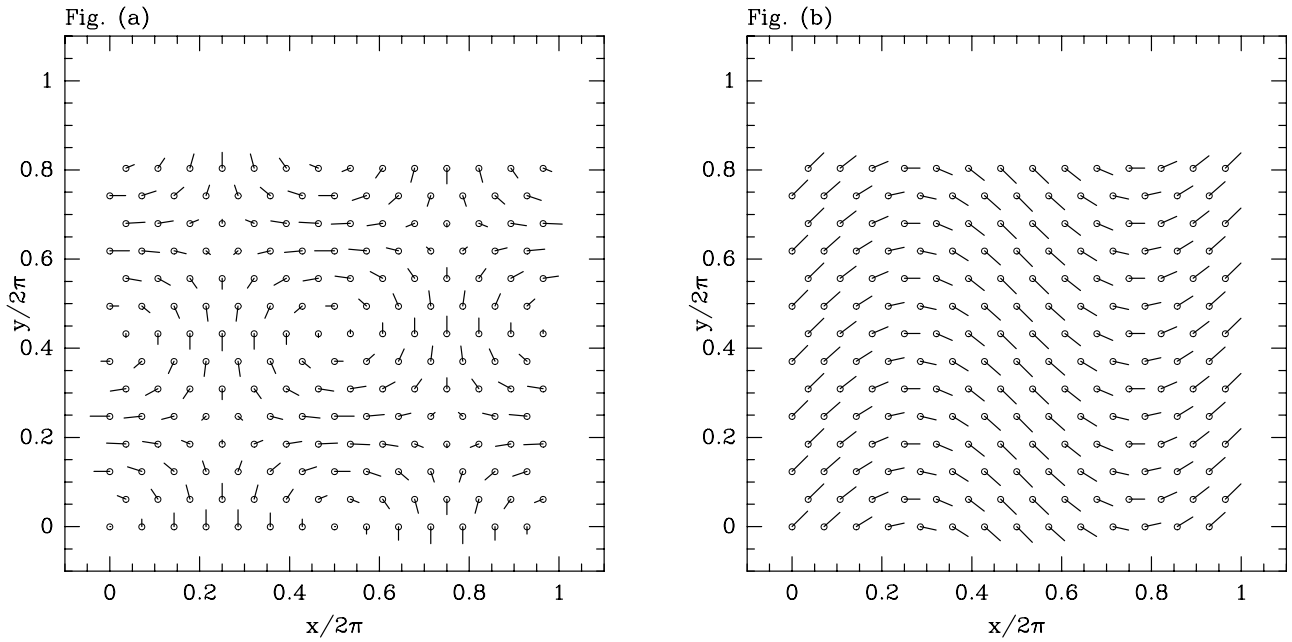

Figure 4-4: The velocity field of the hexagonal Taylor vortex and the hexagonal shear flow are shown in figures (a) and (b) respectively. Both flows have periodic boundary conditions.

viscosity ν given the discretization parameters $\Delta x, \Delta t$. The coefficient w_{32} is chosen to achieve the desired bulk viscosity. In the case of the hybrid method d3q15H, the bulk viscosity of equation 4.61 is chosen equal to the bulk viscosity of the standard collision operator given by equation 4.57.

The following two sections present experimental evidence regarding the accuracy of the hexagonal d2q7 and the orthogonal d2q9 models in initial and in boundary value problems. Experimental results for the three-dimensional d3q15 model are not presented here. However, the algorithm presented above (both d3q15 and d3q15X) has been tested on simple flows, and appears to work correctly. The accuracy of the d3q15 model is expected to be comparable to the accuracy of the d2q9 model.

4.4 Experiments — initial value

First, **initial value problems** are tested. For this purpose, the analytic solutions of a decaying Taylor vortex and a decaying shear flow are used. These flows are two-dimensional and have periodic boundary conditions. Figure 4-4 shows the velocity vector fields of the flows. The decaying Taylor vortex (G.I. Taylor1923 [51]) has the following analytic solution,

$$\begin{aligned} V_x(x,y,t) &= (-1/A)\,\cos(Ax)\,\sin(By)\,\exp(-2\alpha\,\nu t) \\ V_y(x,y,t) &= (1/B)\,\sin(Ax)\,\cos(By)\,\exp(-2\alpha\,\nu t) \\ P(x,y,t) &= -(1/4)\,[\cos(2Ax)/A^2 + \cos(2By)/B^2]\,\exp(-4\alpha\,\nu t) \end{aligned} \tag{4.62}$$

where the constant α is equal to $(A^2 + B^2)/2$, and ν is the kinematic viscosity. The length constants A, B are chosen $A = 1$ and $B = 2/\sqrt{3}$ to produce the **hexagonal Taylor vortex**, and $A = B = 1$ to produce the **orthogonal Taylor vortex**. The former is used to test the hexagonal 7-speed model, and the latter is used to test the orthogonal 9-speed model. The flow region of the hexagonal Taylor vortex is $0 <= x <= 2\pi$ and $0 <= y <= \pi\sqrt{3}$, and can be covered exactly by a hexagonal lattice using periodic boundary conditions. Similarly, the flow region of the orthogonal Taylor vortex is $0 <= x <= 2\pi$ and $0 <= y <= 2\pi$, and can be covered exactly by an orthogonal lattice using periodic boundary conditions.

The decaying shear flow has the following analytic solution,

$$\begin{aligned} V_x(x,y,t) &= A \\ V_y(x,y,t) &= B\,\cos(k\,x - k\,A\,t)\,\exp(-k^2\,\nu\,t) \\ P(x,y,t) &= \text{constant} \end{aligned} \tag{4.63}$$

where the constant k is chosen $k = 1$ so that x varies between $0 <= x <= 2\pi$, and the length constants A, B are chosen $A = B = 1$ so that the horizontal velocity is equal to the maximum vertical velocity. The vertical extent of the shear flow is chosen $0 <= y <= \pi\sqrt{3}$ for the hexagonal case, and $0 <= y <= 2\pi$ for the orthogonal case in complete analogy with the Taylor vortex.

In all of the results reported below, the coefficient of shear viscosity is chosen equal to one, $\nu = 1$. The measured error V^E denotes the velocity relative error, and

is calculated according to the following formula,

$$V^E = \frac{\sum_{x,y} |V_x - V_x^*|}{\sum_{x,y} |V_x^*|} + \frac{\sum_{x,y} |V_y - V_y^*|}{\sum_{x,y} |V_y^*|} \tag{4.64}$$

where V^* denotes the exact analytic solution, and the sums are taken over the whole grid. In the case of the Hagen-Poiseuille flow and the oscillating plate problem (see section 4.5) where $\sum_{x,y} |V_y^*| = 0$, we use a different normalization as follows,

$$V^E = \frac{\sum_{x,y} |V_x - V_x^*| + \sum_{x,y} |V_y - V_y^*|}{\sum_{x,y} |V_x^*|} \tag{4.65}$$

Double-precision arithmetic is used in all of the reported results unless stated otherwise (for example in figure 4-13).

The Mach number M is defined using the maximum fluid speed at time zero, which is equal to 1.0 for all the test cases,

$$M = 1/c_s = \Delta t/(\Delta x\sqrt{3w_0}) \tag{4.66}$$

Also, the pseudo-Mach number or "computational Mach number" M_c is defined,

$$M_c = 1/c = \Delta t/\Delta x \tag{4.67}$$

Below, M_c is used in the figures rather than M because the discretization error of the lattice Boltzmann method depends on M_c rather than M as we will see below. In the case of the Taylor vortex, which is a solution of the incompressible Navier Stokes equations, the compressible effects are kept smaller than the discretization error by choosing $w_0 = 1/7$. Both the compressible effects and the discretization error decrease quadratically with M_c, and the choice $w_0 = 1/7$ keeps the compressible effects smaller than the discretization error in the Taylor vortex at least (see section 4.4.4). In the case of shear flow, which has zero density gradient and is a solution of the compressible Navier Stokes equations, the error is independent of the Mach number M and it depends only on M_c.

For the hexagonal 7-speed model, the choice $w_0 = 1/7$ produces a Mach number that satisfies the relation $M = (1.53\, M_c) = (1.53\Delta t/\Delta x)$. For the orthogonal 9-speed

model, the choice $y_0 = w_0/4$ and $w_0 = 1/7$ produces $M = (1.53\, M_c)$ also. Another choice $w_0 = 10^{-6}/3$ is discussed briefly in section 4.4.3 for the purpose of allowing high Mach numbers with small M_c in particular $M = (10^3\, M_c)$. We also note that different values of w_0 are used in section 4.4.4 for the purpose of examining the error of the lattice Boltzmann method as a function of Δt while keeping the Mach number constant. In particular, the Mach number is kept constant by varying w_0 in proportion to Δt^2 (see equation 4.68). This study allows us to distinguish between compressible effects and the discretization error of the lattice Boltzmann method.

4.4.1 Initialization error

This section compares the different methods of initialization which are described in section 4.2, and are denoted by d2q7F0, d2q7F1, d2q7F1M, and d2q7H. We recall that the simplified first-order Chapman-Enskog expansion (equation 4.37, 4.38) is identical to the hybrid method d2q7H, and thus there is no need to test it separately. Figure 4-5 plots the error during the first 10 steps of the simulation. A 30×30 grid is used ($\Delta x = 2\pi/30 = 0.2094$). Figure (a) plots the error in the case of the hexagonal Taylor vortex, using $\Delta t = 0.001$ which gives $\tau = 0.5912$ for the standard collision operator. The curves shown correspond to d2q7F0, d2q7F1, d2q7F1M, d2q7H (solid, dashed, dotted, dash-dotted lines). Figure (b) plots the same data using $\Delta t = 0.025$ which gives $\tau = 2.780$ for the standard collision operator. We can see that the first-order momentum-conserving Chapman-Enskog expansion d2q7F1M and the hybrid method d2q7H produce very similar results, and they are are the most accurate in all cases. We can also see that the first-order Chapman-Enskog expansion d2q7F1 that does not conserve momentum is more accurate than the zero-order expansion d2q7F0 when $\tau < 1$ and inversely when $\tau > 1$. Figures (c) and (d) plot the same data as figures (a) and (b) for the case of shear flow. The results are qualitatively the same. The experiments demonstrate that the hybrid method can be used to initialize accurately the populations F_i from the fluid variables ρ, V_x, V_y in an initial

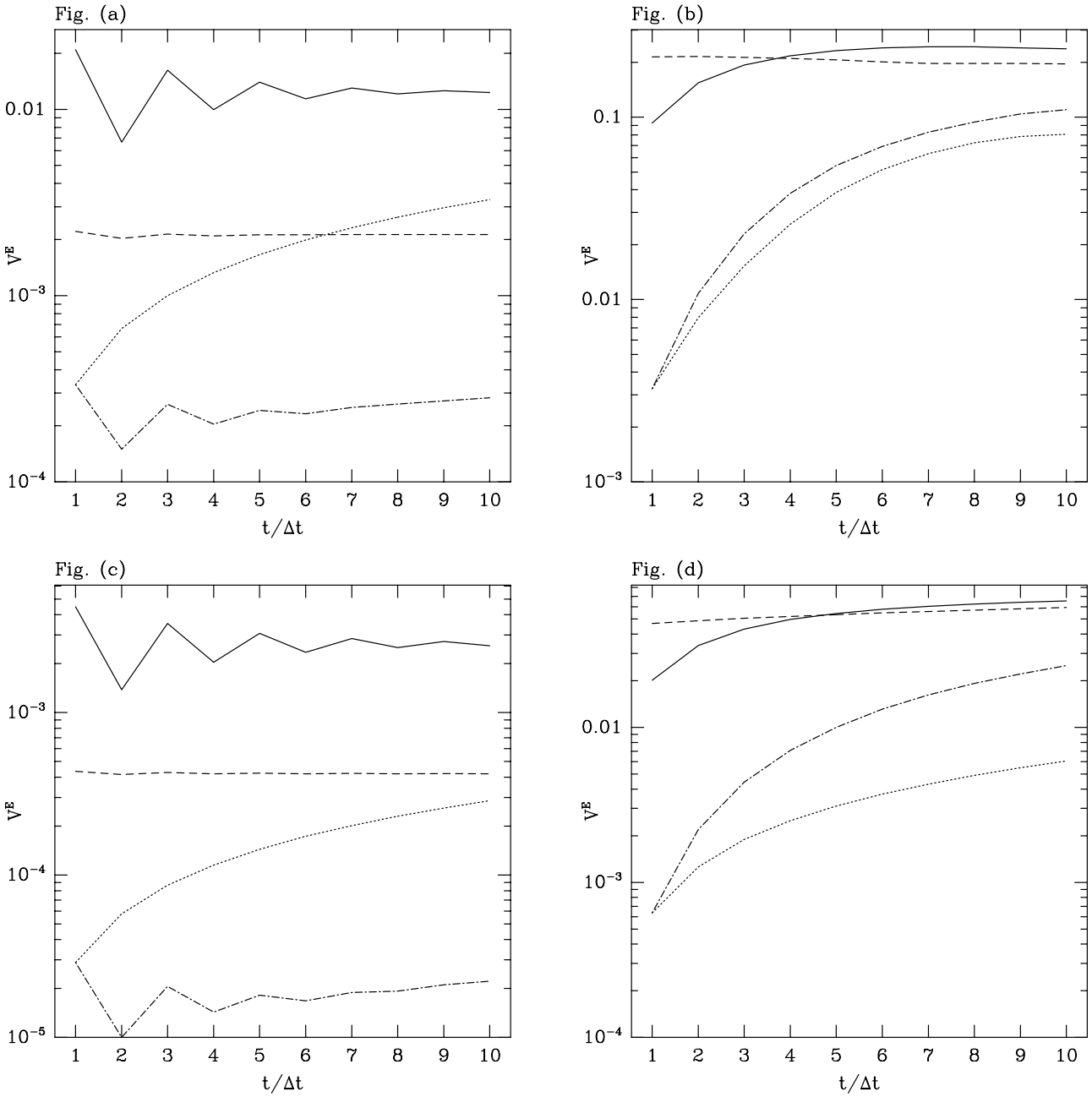

Figure 4-5: The four initialization methods d2q7F0, d2q7F1, d2q7F1M, d2q7H (solid, dashed, dotted, dash-dotted lines) are compared using a 30×30 grid and periodic boundary conditions. Figures (a) and (b) plot the error in simulating the hexagonal Taylor vortex using $\Delta t = 0.001$ and $\Delta t = 0.025$ respectively ($\tau = 0.5912$ and $\tau = 2.780$). Figures (c) and (d) plot the same data in the case of shear flow.

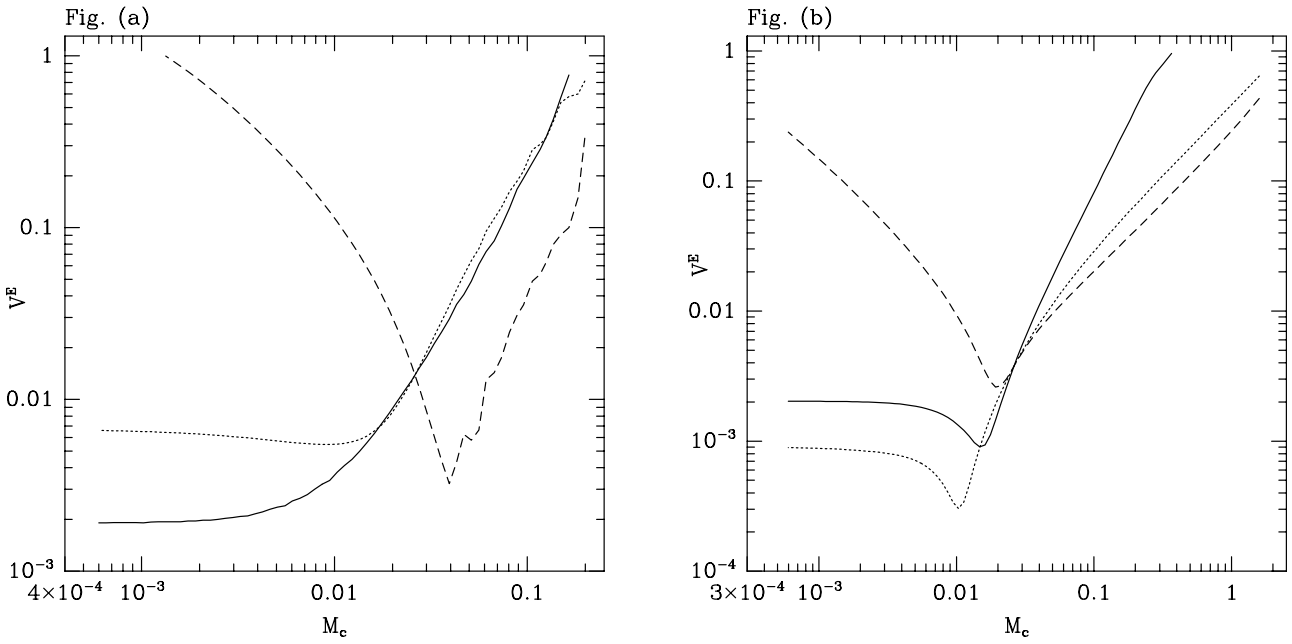

Figure 4-6: The performance of the extended collision operator is shown during repeated iterations. The error is plotted against M_c with Δt varying, and is calculated at the final time $T = 1.0$. The curves correspond to the hybrid method d2q7H, to the extended collision operator d2q7X using finite differences to calculate the gradients, and again to the extended collision operator d2q7X using the known analytic solution to calculate the gradients (solid, dashed, dotted lines). Figure (a) shows the error in simulating the hexagonal Taylor vortex, and figure (b) shows the error in simulating the hexagonal shear flow.

value problem.

4.4.2 Iterating the extended collision operator

This section examines the performance of the extended collision operator when iterated many times. We recall that the extended collision operator uses the gradients of the fluid velocity to control the viscosity. Figure 4-6 shows the error in simulating the hexagonal Taylor vortex and the hexagonal shear flow using a 30×30 grid. The error is plotted against M_c with Δt varying, and is calculated at the final time $T = 1.0$ when the maximum velocity of the hexagonal Taylor vortex is approximately $1/10$ of its initial value. The curves correspond to the hybrid method d2q7H, and to

the extended collision operator d2q7X using finite differences to calculate the gradients, and again to the extended collision operator d2q7X using the exact solution to calculate the gradients (solid, dashed, dotted lines). When the curves of figure 4-6 intersect at $M_c = 0.026$, the relaxation parameter τ of the standard collision operator is equal to one, and the coefficients w_{31}, w_{32}, z_{32} of the extended collision operator vanish (see equation 4.32). At this point, the extended collision operator is identical to the standard collision operator.

As M_c decreases below the value $M_c = 0.026$, the error of the extended collision operator d2q7X using finite differences to calculate the gradients begins to grow and approaches relative error one as M_c goes to zero (dashed line). By contrast, the error of the extended collision operator d2q7X using the analytic solution to calculate the gradients decreases towards a minimum error (dotted line) which is determined by the spatial discretization error of the 30×30 grid. This shows that the use of finite differences creates problems after repeated iterations. As explained in section 4.2 the inexactness of finite differences produces an error in viscosity which accumulates and becomes large after repeated iterations.

The hybrid method d2q7H does not suffer from the problems of the extended collision operator after repeated iterations because the hybrid method uses the standard collision operator at the inner nodes after the first step (all nodes are inner in this experiment). Figure 4-6 shows that the hybrid method performs well in the case of periodic boundary conditions, and remains accurate as M_c goes to zero (solid line). In section 4.5, it is shown that the hybrid method performs well in the case of boundary value problems also.

4.4.3 Comparison with projection method

This section compares the error of the hybrid method d2q7H and the error of an explicit finite difference projection method in simulating the hexagonal Taylor vortex and the hexagonal shear flow with periodic boundary conditions. Both of these flows

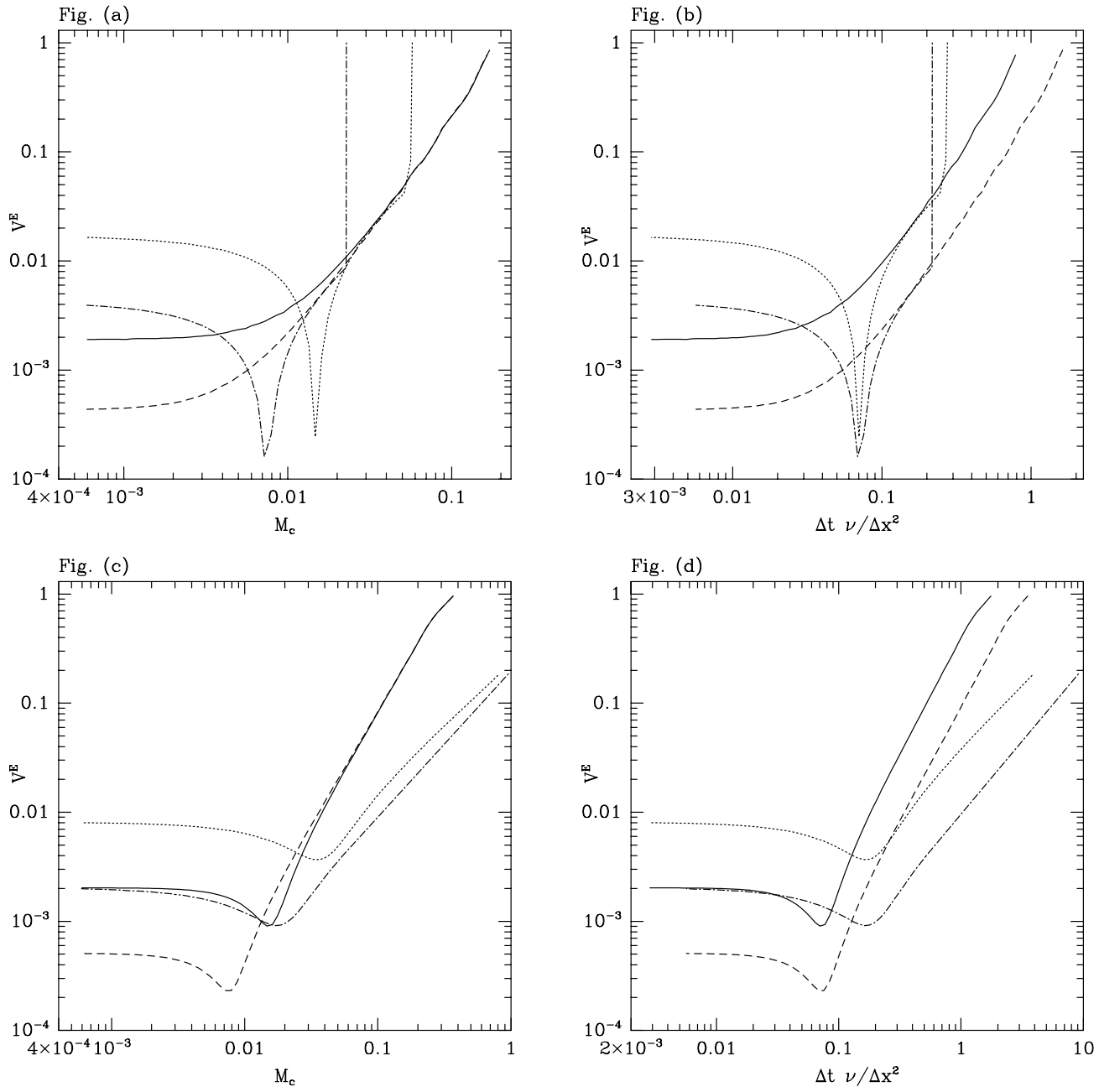

Figure 4-7: The error of the lattice Boltzmann method d2q7H is compared against the error of the explicit finite difference projection method EP7. The curves correspond to d2q7H using 30×30 grid, d2q7H using 60×60 grid, EP7 using 30×30 grid, and EP7 using 60×60 grid (solid, dashed, dotted, dash-dotted lines). Figures (a) and (b) show the error in simulating the hexagonal Taylor vortex, and figures (c) and (d) show the error in simulating the hexagonal shear flow.

are defined in the hexagonal region $0 <= x <= 2\pi$ and $0 <= y <= \pi\sqrt{3}$, which means that the finite difference projection method must use the discretization $\Delta y = \Delta x\sqrt{3}/2$. Below, we refer to the projection method with the symbol EP7 when it is applied to a hexagonal region, and with the symbol EP9 when it is applied to an orthogonal region (this is done in later sections). The explicit finite difference projection method is described in section 3.4.

Figure 4-7 (a) plots the error in simulating the hexagonal Taylor vortex against M_c with Δt varying. The error is calculated at the final time $T = 1.0$ when the maximum velocity of the hexagonal Taylor vortex is approximately 1/10 of its initial value. The curves correspond to d2q7H using 30×30 grid, d2q7H using 60×60 grid, EP7 using 30×30 grid, and EP7 using 60×60 grid (solid, dashed, dotted, dash-dotted lines). Figure (b) plots the same data against the dimensionless ratio $\Delta t\,\nu/\Delta x^2$ which facilitates comparison between different grids. Figures (c) and (d) plot the same data for shear flow. We can see that the Taylor vortex triggers an instability in the explicit projection method EP7 when $\Delta t\,\nu/\Delta x^2 >= 0.2$, but the shear flow does not trigger any instability.

With regard to the lattice Boltzmann method, we observe that it fails to approximate the solution (has a relative error of 1.0) when M_c is larger than 0.2 approximately. In the case of the Taylor vortex, which is a solution of the incompressible fluid flow equations, it may appear that the problem arises from the compressibility of the lattice Boltzmann fluid (when $M_c \sim 0.2$, the Mach number is approximately $M = 1.53\,M_c = 0.3$). In the case of the shear flow, however, compressibility is not important. The shear flow is a solution of the compressible fluid flow equations, and it should be easily computed by the lattice Boltzmann method both at low and high Mach numbers. In fact, the shear flow can be computed easily at high Mach numbers by using a smaller w_0, for example $w_0 = 10^{-6}/3$ (see below).

The limitations of the lattice Boltzmann method shown in figure 4-7 when M_c is larger than 0.2 persist independent of the Mach number. The limitations arise

because the microscopic speed $\Delta x/\Delta t$ becomes comparable to the fluid speed when M_c approaches 1.0, and the high-order terms in the Chapman-Enskog expansion (which are neglected in deriving the Navier Stokes equations) become significant, and produce behavior that differs from the Navier Stokes equations.

With regard to simulating shear flow at high Mach numbers, we can choose $w_0 = 10^{-6}/3$ which gives $M = 10^3\, M_c$. The error of the lattice Boltzmann method d2q7H in simulating shear flow with $M = 10^3\, M_c$ is identical to the error plotted in figure 4-7(c). The error in simulating shear flow is independent of the Mach number because the density gradients are zero everywhere.

4.4.4 Quadratic convergence

This section shows that the lattice Boltzmann method has second-order convergence both in space and in time. Second-order convergence in space means that the error decreases quadratically with Δx while keeping the dimensionless ratio $\Delta t\, \nu/\Delta x^2$ constant (Fletcher [18, p.75]). Second-order convergence in time means that the error decreases quadratically with Δt while keeping the space discretization Δx constant. Furthermore, we are interested in the true discretization error and not the error that arises from compressibility. When using a compressible fluid code such as the lattice Boltzmann method to simulate incompressible flow such as the Taylor vortex, it is important to distinguish between the error that arises from compressibility and the error that arises from finite discretization.

In figure 4-7 the Mach number decreases in proportion to M_c, and thus the effects of compressibility and finite discretization can not be distinguished without further analysis. To distinguish between the effects of compressibility and discretization error, we perform the same simulations as those in figure 4-7, while keeping the Mach number constant and varying the density coefficient w_0 as follows,

$$w_0 \;=\; \frac{1}{3}\left(\frac{\Delta t}{\Delta x\, M}\right)^2 \tag{4.68}$$

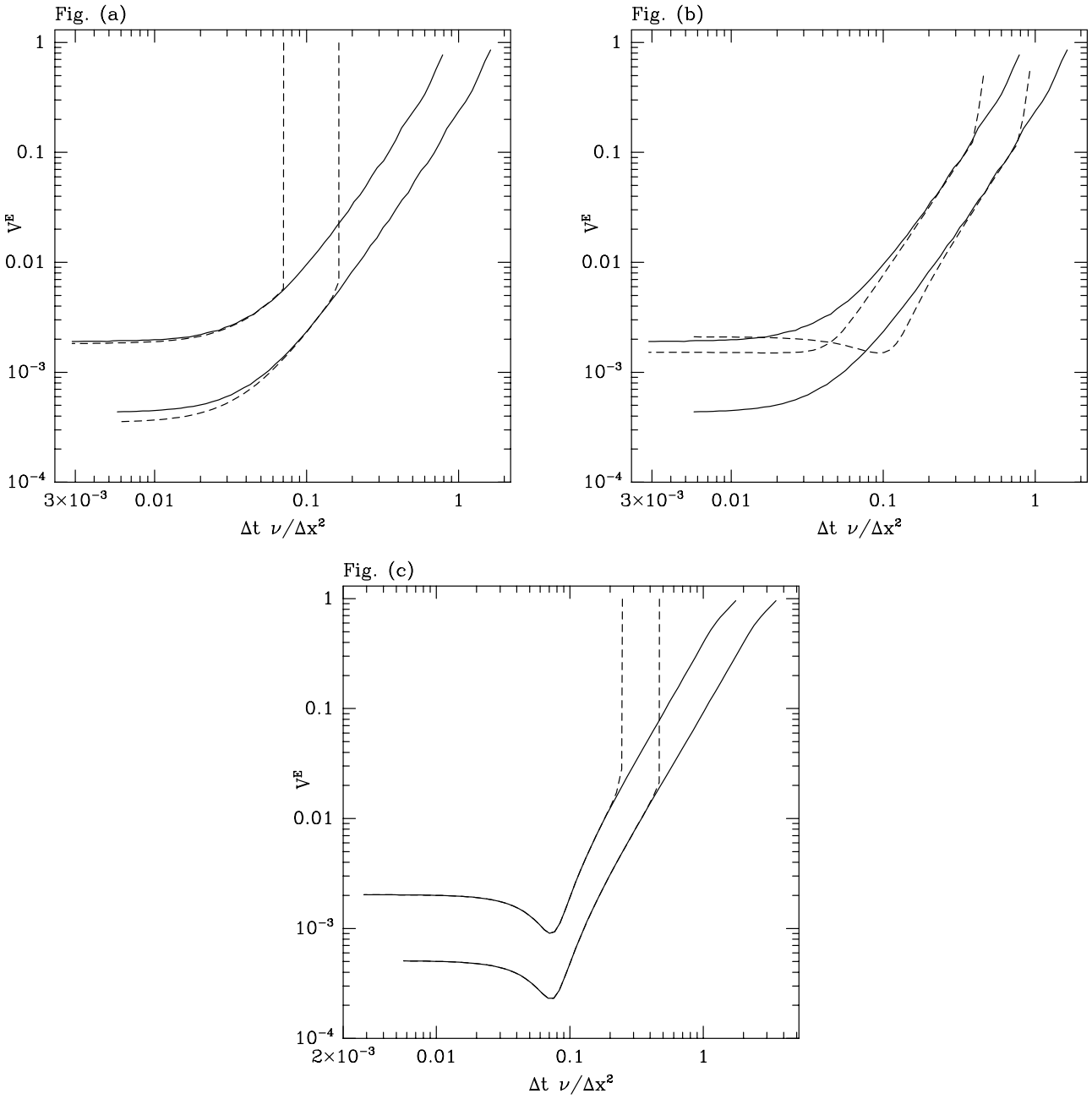

Figure 4-8: The error of d2q7H is plotted against M_c with Δt varying, while keeping the Mach number M constant and varying the density parameter w_0 (two dashed lines). For comparison purposes, the error of d2q7H when the Mach number varies and the density parameter $w_0 = 1/7$ is held constant is also shown (two solid lines). Results are shown for a 30×30 and a 60×60 grid. Figures (a), (b), (c) correspond to the hexagonal Taylor vortex at $M = 0.02$, the hexagonal Taylor vortex at $M = 0.1$, and the hexagonal shear flow at $M = 0.05$ respectively.

In figure 4-8 (a), we show the error of d2q7H in simulating the hexagonal Taylor vortex at constant Mach number $M = 0.02$ using a 30×30 grid and a 60×60 grid (two dashed lines). For comparison purposes, we also show the error of d2q7H using constant $w_0 = 1/7$ and variable Mach number $M = 1.53\,M_c$ (two solid lines). The constant Mach number curves are identical to the constant w_0 curves except for instabilities which are discussed below. This indicates that the compressible effects at Mach number $M = 0.02$ are smaller than the discretization error of both the 30×30 and 60×60 grids. The instability of the constant Mach number curves (dashed lines) is expected and it occurs when the density coefficient w_0 given by equation 4.68 becomes greater than $1/6$ which forces the density coefficient z_0 to become negative. Similar instabilities can be seen in figure 4-8 (c) which plots the same experiment for shear flow at constant Mach number $M = 0.05$.

It is important to note that if we keep the Mach number constant while decreasing the grid spacing (Δx), then a sufficiently fine grid will eventually bring out the compressible effects. For example, figure 4-8 (b) shows the same data as figure 4-8 (a) while keeping the Mach number constant at $M = 0.1$. In the case of the 30×30 grid the constant Mach number curves are identical to the constant w_0 curves as before, which indicates that the discretization error of the 30×30 grid is larger than the compressible effects of Mach number $M = 0.1$. In the case of 60×60 grid however, the constant Mach number curves reach a minimum error (as Δt goes to zero) that is much greater than the minimum error of the constant w_0 curves. This is because the discretization error of the 60×60 grid becomes smaller than the compressible effects of Mach number $M = 0.1$ when $\Delta t\,\nu/\Delta x^2$ becomes smaller than 0.1 approximately.

In general, we can calculate the Mach number at which compressible effects become larger than the discretization error of any grid by doing more numerical experiments of the kind shown in figure 4-8. Such a study is not necessary for our purposes however. Figures 4-8(a) and 4-8(b) are enough to show that the compressible effects in simulating the Taylor vortex are smaller than the discretization error of the 30×30

and 60×60 grids when w_0 is constant and the Mach number varies as $M = 1.53\, M_c$. Accordingly, we can examine the error curves of figure 4-8 and also of figure 4-7 to find out how the discretization error of the lattice Boltzmann method decreases with finer resolution.

If we examine the logarithmic plots of figure 4-7, we see that the error decreases quadratically with Δt (it has a slope of -2) until a minimum spatial discretization error is reached. In addition the error decreases by a factor of 4 when we go from the 30×30 grid to the 60×60 grid while keeping the dimensionless ratio $\Delta t\, \nu / \Delta x^2$ constant, see figures 4-7 (b) and 4-7 (d). In other words the lattice Boltzmann method has second-order convergence both in space and in time. In section 4.5 we will verify the second-order convergence for boundary value problems also. The explicit finite difference projection method EP7 has first-order convergence in time and second-order convergence in space. The first-order convergence in time of the projection method EP7 can be seen most easily in figures 4-7 (c) and 4-7 (d).

4.4.5 7-speed versus 9-speed

Here, the accuracy of the hexagonal 7-speed model is compared against the accuracy of the orthogonal 9-speed model. Figure 4-9 shows the error of d2q7H applied to the hexagonal Taylor vortex, and the error of d2q9H applied to the orthogonal Taylor vortex (solid and dashed lines). In addition, the error of the explicit finite difference projection method is shown when the projection method is applied to the hexagonal Taylor vortex with $\Delta y = \Delta x \sqrt{3}/2$, and also to the orthogonal Taylor vortex with $\Delta y = \Delta x$ (dotted and dash-dotted lines). A 30×30 grid is used, and the error is calculated at the final time $T = 1.0$. We can see that the explicit finite difference projection method performs similarly on the hexagonal and the orthogonal Taylor vortices. By contrast, the orthogonal 9-speed model d2q9H is significantly more accurate than the hexagonal 7-speed model d2q7H on this specific problem.

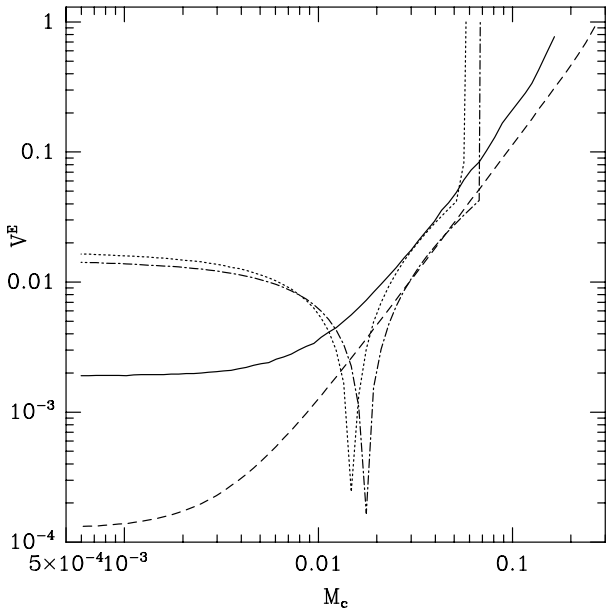

Figure 4-9: The error of d2q7H applied to the hexagonal Taylor vortex, and the error of d2q9H applied to the orthogonal Taylor vortex are shown (solid and dashed lines). In addition the error of the explicit finite difference projection method is shown when applied to the hexagonal Taylor vortex with $\Delta y = \Delta x\sqrt{3}/2$ and also the orthogonal Taylor vortex with $\Delta y = \Delta x$ (dotted and dash-dotted lines).

4.5 Experiments — boundary value

In this section, the orthogonal 9-speed hybrid model d2q9H is tested on boundary value problems with exact solutions, and is also compared against the explicit finite difference projection method EP9. In all of the test cases examined here, both the density and the velocity values are specified exactly at the boundary. The question of how to compute the density at a boundary (such as a non-slip wall) using the computed solution is discussed later in section 4.6.1.

The boundary value problems are the one-quarter Taylor vortex, the Hagen-Poiseuille flow, and the oscillating plate above a stationary wall. Figure 4-10 shows the velocity vector fields of these flows, and also indicates the boundary nodes of each flow by drawing a square around the boundary nodes. Figure 4-10 (c) is plotted at time $t = 0.4$ when the oscillating plate starts moving to the left while the fluid below is still moving to the right.

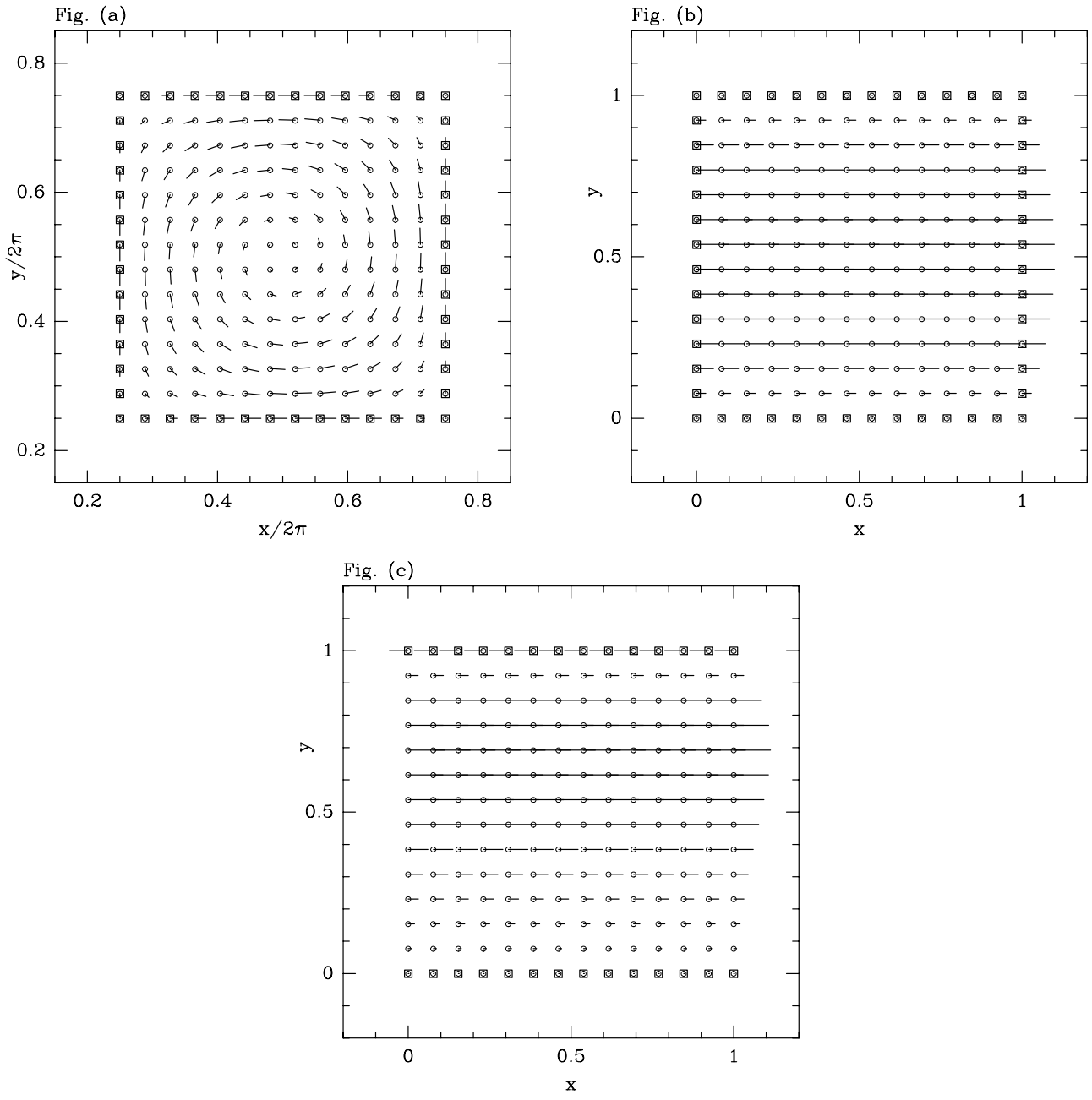

Figure 4-10: The velocity field of the one-quarter Taylor vortex, the Hagen-Poiseuille flow, and the oscillating plate problem are shown in figures (a), (b), (c) respectively. Boundary nodes are marked with a square. Figure (c) is plotted at time $t = 0.4$ when the oscillating plate starts moving to the left and the fluid below is still moving to the right.

The one-quarter Taylor vortex is defined in the region $\pi/2 <= x <= 3\pi/2$ and $\pi/2 <= y <= 3\pi/2$. The exact solution is given by equation 4.62 with $A = B = 1$. The velocity and pressure are specified at the boundary by evaluating the exact solution at the horizontal and vertical lines $\pi/2 <= x <= 3\pi/2$ and $\pi/2 <= y <= 3\pi/2$. From the pressure, we calculate the density using equation 4.33.

The Hagen-Poiseuille flow is defined in the region $0 <= x <= 1$ and $0 <= y <= 1$. The analytic solution is as follows,

$$\begin{aligned} V_x(x,y,t) &= -(y^2 - y)\,\Delta P \,/\, (2\nu) \\ V_y(x,y,t) &= 0 \\ P(x,y,t) &= (0.5 - x)\,\Delta P \end{aligned} \tag{4.69}$$

The pressure gradient ΔP is chosen $\Delta P = (8.0\nu)$ so that the maximum fluid speed is 1.0 when $y = 1/2$. The velocity and the density are specified at the boundary by evaluating the exact solution at $0 <= x <= 1$ and $0 <= y <= 1$.

The oscillating plate problem is defined in the region $0 <= x <= 1$ and $0 <= y <= 1$ with periodic boundary conditions in the horizontal direction $x = 0$ and $x = 1$. The velocity is specified at the top and bottom plates by evaluating the exact solution, namely,

$$\begin{aligned} y = 1: &\qquad V_x = cos(\omega\, t) \qquad V_y = 0 \\ y = 0: &\qquad V_x = 0 \qquad\qquad\;\; V_y = 0 \end{aligned} \tag{4.70}$$

The density at the top and bottom plates is set equal to 1.0 (the exact solution has constant pressure everywhere). The frequency of oscillation ω is chosen $\omega = 20$ so that the oscillating plate executes 3.18 cycles of oscillation during the time interval $T = 1.0$ which is used for testing (this is an arbitrary choice). The analytic solution

of the oscillating plate problem (see section 2.5.3) is given by the following equations,

$$\begin{aligned} V_x(x,y,t) &= (\cosh A \sin A(-2\cosh B \sin B \cos \omega t + 2\cos B \sinh B \sin \omega t) \\ &\quad - \cos A \sinh A(2\cosh B \sin B \sin \omega t + 2\cos B \sinh B \cos \omega t)) \\ &\quad / \ (\cos 2B - \cosh 2B) \qquad (4.71) \\ V_y(x,y,t) &= 0 \\ P(x,y,t) &= \text{constant} \end{aligned}$$

where $A = y\sqrt{\omega/(2\nu)}$ and $B = \sqrt{\omega/(2\nu)}$, and ν is the kinematic viscosity.

In the case of steady flow such as the Hagen-Poiseuille flow, we initialize the variables ρ, V_x, V_y equal to the exact steady state solution. Then, we iterate for 100 steps, and test whether the fluid is in steady state. If the fluid is in steady state, we measure the velocity relative error V^E. Otherwise, we keep iterating until the fluid reaches steady state. The goal of this procedure is to measure the error at steady state and not to characterize how quickly the fluid reaches steady state. The criterion for steady state is that the relative change in velocity between successive iterations divided by Δt must be less than 10^{-6},

$$\frac{\sum_{x,y} |V_x(t+\Delta t) - V_x(t)|}{\sum_{x,y} |V_x^*|} \ < \ 10^{-6} \ \ \Delta t \qquad (4.72)$$

and similarly for V_y.

In the case of transient flow such as the one-quarter Taylor vortex and the oscillating plate, the error V^E is measured at the final time $T = 1.0$ using equations 4.64 and 4.65.

4.5.1 Comparison between LB boundary schemes

The hybrid method d2q9H uses the standard collision operator at the inner nodes, and the extended collision operator at the boundary nodes. An important issue is the calculation of the gradients of the fluid velocity at the boundary nodes. The best results are achieved when the gradients of the fluid velocity are specified using the

exact solution. In practice, however, the velocity gradients at the boundary nodes are usually not known. For example, the gradient $\partial V_x/\partial y$ at the top and bottom walls of the driven cavity problem (not reported here but see Peyret&Taylor [38, p.199]) can not be specified because it is part of the solution that we seek to compute. When a velocity gradient can not be specified, finite differences must be used to estimate it.

In the experiments below, different ways of specifying the velocity gradients at the boundary are tested. First, the exact solution is used to specify all of the velocity gradients at the boundary nodes. Second, finite differences are used to estimate all the velocity gradients at the boundary nodes. When the exact solution is used, the method is denoted by d2q9H$_{XD}$ (XD stands for exact derivatives at the boundary). When first-order asymmetric differences are used, the method is denoted by d2q9H$_{1FD}$. When second-order asymmetric differences are used, the method is denoted by d2q9H$_{2FD}$.

In the experiments below, we also test the lattice Boltzmann scheme d2q9F0 which uses the standard collision operator at every node, both boundary and inner nodes. At the boundary nodes, the method d2q9F0 sets the populations F_i equal to the equilibrium values F_i^{eq} of the standard collision operator given by equation 4.13. At startup, the method d2q9F0 normally initializes the F_i equal to the equilibrium values F_i^{eq} of the standard collision operator. In the present section, however, the extended collision operator is used for initialization in order to avoid initial errors, and the standard collision operator is used after the first step.

Regarding boundary conditions for the explicit finite difference projection method, the velocity at the boundary is specified from the exact solution, and the pressure P is specified from the requirement $\partial P/\partial n = 0$ at the boundary, where ∂n denotes the direction normal to the boundary (Peyret&Taylor [38, p.160]). The condition $\partial P/\partial n = 0$ is applied at the beginning of the SOR calculation using the values of P at the previous time step, and the resulting boundary values for the pressure P are held constant throughout the SOR calculation.

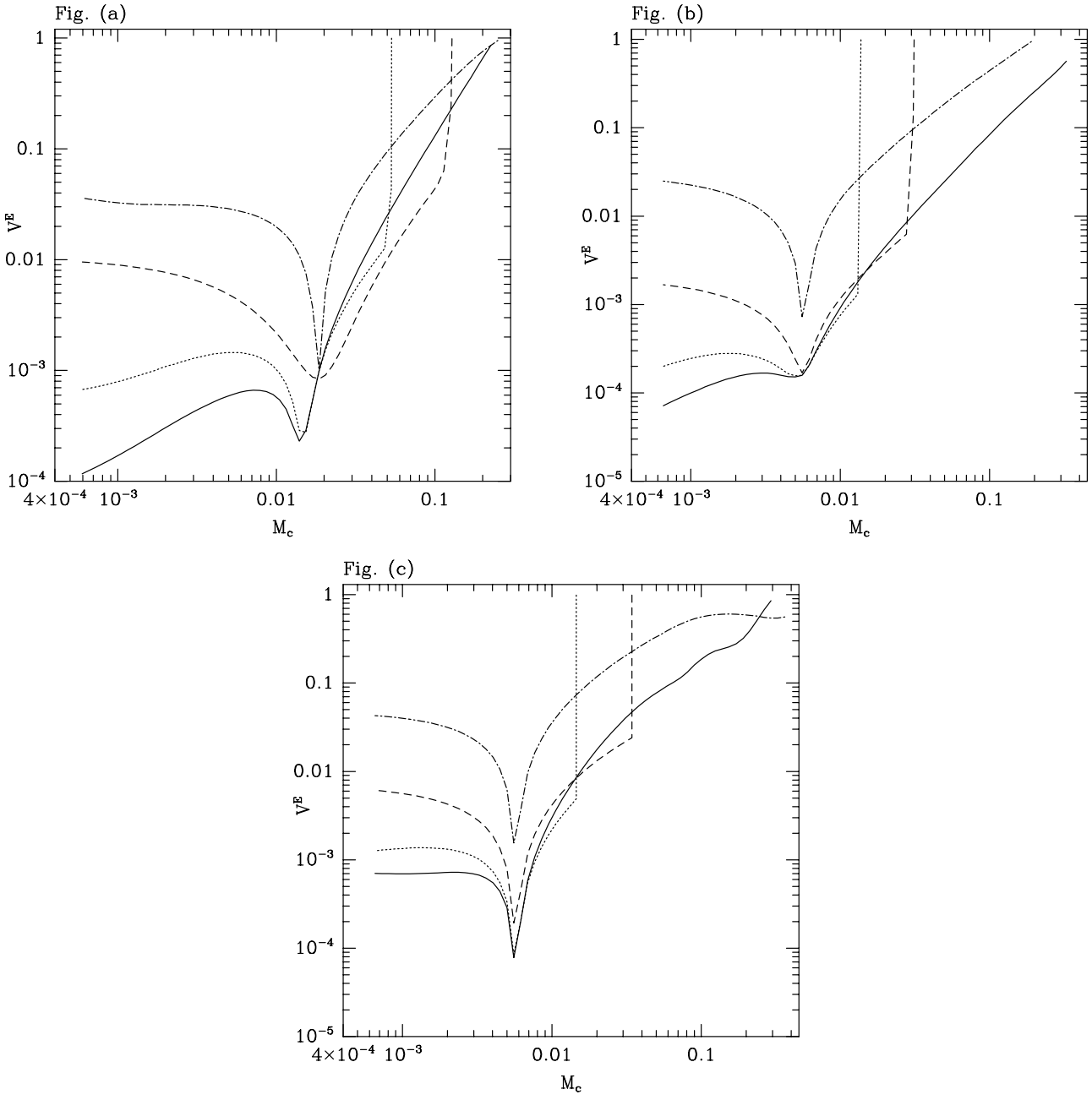

Figure 4-11: The error of d2q9H$_{XD}$, d2q9H$_{1FD}$, d2q9H$_{2FD}$, and d2q9F0 (solid, dashed, dotted, and dash-dotted lines) is shown in simulations of the one-quarter Taylor vortex, the Hagen-Poiseuille flow, and the oscillating plate — figures (a), (b), (c) respectively.

Figure 4-11 compares the methods d2q9H$_{XD}$, d2q9H$_{1FD}$, d2q9H$_{2FD}$, and d2q9F0 (solid, dashed, dotted, and dash-dotted lines) in simulations of the one-quarter Taylor vortex, the Hagen-Poiseuille flow, and the oscillating plate, figures (a), (b), (c) respectively. A 30×30 grid is used, and the error is plotted against M_c with Δt varying, and is calculated at the final time $T = 1.0$. We can see that the standard collision operator d2q9F0 achieves smallest error when the relaxation parameter $\tau = 1$, at which point the standard and extended collision operators are identical. We can also see that the hybrid method achieves best results when the velocity gradients at the boundary nodes are specified from the exact solution (method d2q9H$_{XD}$). Further, we can see that the finite differences at the boundary (d2q9H$_{1FD}$ and d2q9H$_{2FD}$) trigger instabilities when M_c becomes large, and that first-order differences are a little more stable than second-order differences. However, second-order differences are recommended because they are more accurate with regard to the error in pressure (which is not shown here, but see page 225). As explained on page 140, all the numerical tests of this chapter examine the error in velocity only.

4.5.2 Comparison with incompressible finite differences

Figure 4-12 compares the error of the lattice Boltzmann method d2q9H$_{XD}$ against the error of the incompressible finite difference projection method EP9 in simulations of the one-quarter Taylor vortex, the Hagen-Poiseuille flow, and the oscillating plate, figures (a), (b), (c) respectively. The error is plotted against the dimensionless ratio $\Delta t\, \nu / \Delta x^2$ to facilitate comparison between different grids. The curves correspond to d2q9H$_{XD}$ using 30×30 grid, d2q9H$_{XD}$ using 60×60 grid, EP9 using 30×30 grid, and EP9 using 60×60 grid (solid, dashed, dotted, dash-dotted lines). Figure (b) shows most clearly the rate of convergence in time. The lattice Boltzmann method has second-order convergence in time (slope -2), and the finite difference method EP9 has first-order convergence in time (slope -1). Both methods have second-order convergence in space.

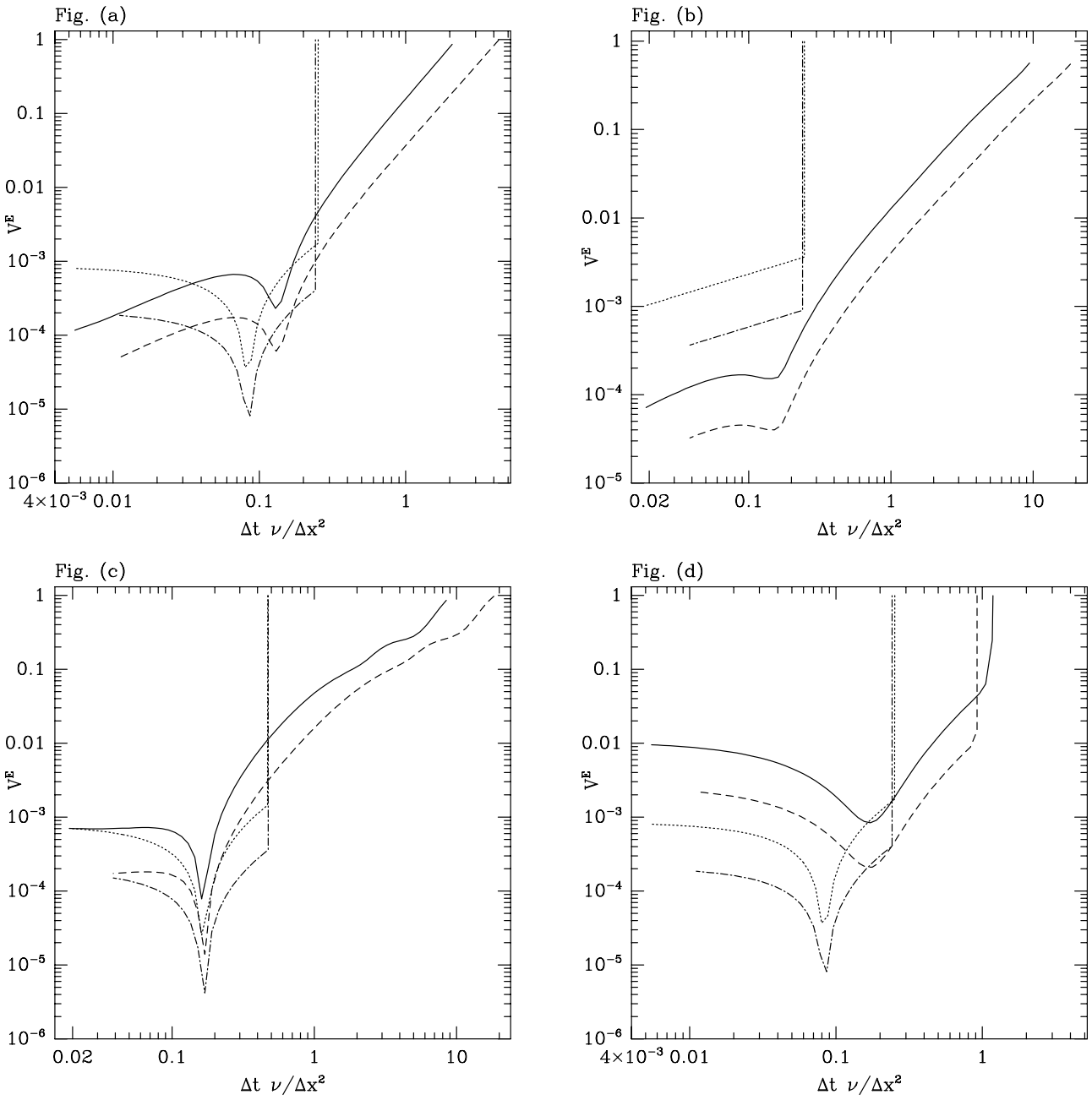

Figure 4-12: The error of the lattice Boltzmann method d2q9H$_{XD}$ is compared against the error of the incompressible finite difference method EP9. The curves correspond to d2q9H$_{XD}$ using 30×30 grid, d2q9H$_{XD}$ using 60×60 grid, EP9 using 30×30 grid, and EP9 using 60×60 grid (solid, dashed, dotted, dash-dotted lines). Figures (a), (b), (c) show simulations of the one-quarter Taylor vortex, the Hagen-Poiseuille flow, and the oscillating plate respectively. Figure (d) shows the same experiment as figure (a) using d2q9H$_{1FD}$ instead of d2q9H$_{XD}$.

It is worth noting that the lattice Boltzmann method has second-order convergence overall even when first-order differences are used to calculate the velocity gradients at the boundary nodes. This can be seen in figure 4-12 (d) which corresponds to the same experiment as figure 4-12 (a) but uses the method d2q9H$_{1FD}$ instead of the method d2q9H$_{XD}$.

4.6 More on boundary conditions

4.6.1 Density calculation at non-slip wall

The modeling of a non-slip wall using the lattice Boltzmann method is discussed here. Suitable boundary conditions must insure that the velocity components V_x, V_y vanish at a non-slip wall, and further that there is a way of calculating the density at a non-slip wall. There are basically two approaches of imposing boundary conditions at a non-slip wall using the lattice Boltzmann method. The first approach is the traditional bounce-back of the populations, which was in section 4.2.1. The second approach, which is used in the simulations of flue pipes, employs the extended collision operator of section 4.2.

The traditional approach is to bounce-back the populations F_i which are moving outwards, so as to produce incoming populations. As stated earlier in section 4.2.1, the approach of bounce-back leads to a non-slip wall which is located somewhere beyond the last set of nodes of the grid, usually a distance of $\Delta x/2$ away. However, the exact location of the wall is not known, and may vary with the flow conditions near the boundary (Cornubert&et al. [12], Ginzbourg&Adler [21]). Regarding the density, the calculation of density at the wall is not an issue because there are no fluid nodes located on the non-slip wall. The density at the nodes nearest the wall is performed in the same way as for all the interior nodes.

The second approach of modeling a non-slip wall, which is used in the simulations of flue pipes, employs the extended collision operator of section 4.2 together with

bounce-back as follows. First, a bounce-back of the outgoing populations is performed in order to produce incoming populations which are used to calculate the density at the wall. [5] Then, the extended collision operator is applied at the nodes of the wall using $V_x = V_y = 0$. The gradients of V_x, V_y which are needed by the extended collision operator are calculated using finite differences. The benefit of using the extended collision operator at the boundary nodes is that the non-slip wall is located precisely at the boundary nodes within numerical error.

4.6.2 Composite grid for lattice Boltzmann

This section outlines how to implement composite grids for the lattice Boltzmann method using the extended collision operator.

The extended collision operator can be used to join a lattice Boltzmann grid with a finite difference grid of the same resolution. For this purpose, a single layer of overlapping nodes must be used. At the overlapping nodes, the future values of ρ, V_x, V_y are calculated using the finite difference method. Subsequently, the future values of ρ, V_x, V_y (already calculated by finite differences) are used to initialize populations F_i at the overlapping nodes, which are used as boundary conditions for the lattice Boltzmann method on the other side of the grid.

The scheme for a composite grid is as follows. Let us assume that lattice Boltzmann is used on a coarse grid at the left side. Going from left to right, there is a point where we change from lattice Boltzmann to finite differences. Further on, the resolution of the finite difference grid is changed to a finer resolution. For simplicity, let us assume that the resolution on the right side is twice the resolution on the left side. Traditional interpolation can be used to join the two finite difference grids of different resolution. Further on, as we move to the right, we change from finite

[5] In my earlier paper [48], I suggested that the density at a wall should be calculated as the average of the populations that "bring fluid into the boundary node" from inner nodes and other neighboring boundary nodes. Further numerical experiments, however, indicate that the average of the populations after bounce-back, which is recommended above, is a slightly better approach.

differences to lattice Boltzmann at the fine resolution.

An issue to remember is that the speed of sound in the lattice Boltzmann method is proportional to $\sqrt{w_0}\Delta x/\Delta t$ where w_0 is a density parameter. Therefore, if the spacing Δx is halved, the density parameter must be divided by 4, or the time step Δt must be halved also. In the former case, the same time step is used globally, and is determined by the finest grid or the smallest spacing Δx. In the latter case, some computation is saved in the coarse grid. In particular, twice as many steps are performed at the finer resolution grid than at the coarser grid. This must be taken into account in the transition region where finite differences and interpolation are used. Presumably, the coarse-grid values can be held constant every other "fine" step of the fine grid.

The transition between grids of different resolution inevitably introduces some error. The desired goal is that the transition error (interpolation error, etc) should not be larger than the error difference between the fine and the coarse grid. This must be tested especially with regard to the propagation of acoustic waves.

Finally, we might wonder why switch back and forth between lattice Boltzmann and finite differences, why not stay with finite differences all the time. The answer is that lattice Boltzmann may provide better stability properties, better handling of boundary conditions, and better modeling of acoustic waves. These issues need to be investigated further in the future.

4.7 Appendix

4.7.1 Roundoff error of lattice Boltzmann

In this section, the numerical roundoff error of the lattice Boltzmann method is discussed using the 7-speed hexagonal LB model for simplicity. It is shown that the roundoff error in the equilibrium population formulas can cause problems under certain conditions. In particular, it is shown that the roundoff error increases as

the ratio V/c becomes smaller (namely, as the Mach number becomes smaller), or as the ratio $\Delta x/\Delta t$ becomes larger. The increasing roundoff error is undesirable because large values of $\Delta x/\Delta t$ are useful for improving the accuracy (reducing the discretization error) of the lattice Boltzmann method. Fortunately, double-precision arithmetic mitigates the roundoff error to a large extent.

Let us consider the implementation of the lattice Boltzmann method according to the equations 4.8,4.10,4.13. The roundoff error (numerical loss of precision) arises in the computation of the equilibrium populations using equation 4.13. This formula is a sum of four terms. If we factor out the density $\rho(\vec{x},t)$, the first term is a constant coefficient w_0 and the remaining terms are proportional to V/c, $(V/c)^2$, and $(V/c)^2$ respectively (see table 4.1). Consequently when V/c is small, for example $V/c \simeq 10^{-3}$, the terms to be added have very disparate sizes and their sum suffers a significant loss of accuracy when the computer aligns the numbers to be added (about 5 or 6 decimal places when $V/c \simeq 10^{-3}$). If single-precision arithmetic is used (about eight decimal places), then the loss of five digits is a serious problem.

term	w_0	w_1	w_{20}	w_{21}
size	1	V/c	$(V/c)^2$	$(V/c)^2$

Table 4.1: The terms of the equilibrium population formula have different sizes. When they are added together, numerical roundoff error can be significant.

Below, numerical experiments are described based on single-precision computer arithmetic, which indicate that the error of the lattice Boltzmann method decreases at first as the speed $\Delta x/\Delta t$ increases, but after some point the error starts to increase with larger $\Delta x/\Delta t$. For example in the Taylor vortex when the maximum fluid speed is 1.0, the error starts to increase at the rate of $(\Delta x/\Delta t)^{1.4}$ when $(\Delta x/\Delta t)$ is larger than 300. Fortunately, the error growth disappears when double-precision arithmetic is used, and this confirms that the breakdown of the method is caused by roundoff error.

An approximate estimate of the extent of roundoff problems is that increasing the ratio $\Delta x/\Delta t$ by a factor of 10 increases the roundoff error in the equilibrium populations by a decimal digit. Therefore, double-precision arithmetic provides roundoff-free operation with ratios $\Delta x/\Delta t$ which are 10^7 times larger than the corresponding ratios in single-precision arithmetic. Clearly, this is a very wide margin for practical calculations.

Apart from using double-precision arithmetic, there is an algebraic transformation which reduces the roundoff error in the equilibrium populations, and it can be used in all cases because it does not involve any additional cost. The algebraic transformation does not eliminate the roundoff error however, and double-precision arithmetic remains necessary. The idea is to modify the populations F_i defined by equations 4.8, 4.10, 4.13 as follows,

$$\begin{aligned} \widehat{F_i} &= F_i - w_0 <\rho> \\ \widehat{F_i^{\rm eq}} &= F_i^{\rm eq} - w_0 <\rho> \end{aligned} \tag{4.73}$$

where the spatial average density $<\rho>$ is constant in time and typically equal to one. The non-moving population become $\widehat{F_0} = F_0 - z_0 <\rho>$. The conservation relations are modified accordingly,

$$\begin{aligned} \rho(\vec{x},t) &= \textstyle\sum_{i=0}^{6} \widehat{F_i}(\vec{x},t) + <\rho> \\ \rho(\vec{x},t)\,\vec{V}(\vec{x},t) &= \textstyle\sum_{i=1}^{6} \widehat{F_i}(\vec{x},t)\,\vec{e}_i \end{aligned} \tag{4.74}$$

The new equilibrium population formulas are as follows,

$$\begin{aligned} \widehat{F_i^{\rm eq}}(\vec{x},t) &= w_0\,(\rho(\vec{x},t) - <\rho>) + \\ &\quad \rho(\vec{x},t)\,\left[w_1(\vec{e}_i\cdot\vec{V}) + w_{20}(\vec{e}_i\cdot\vec{V})(\vec{e}_i\cdot\vec{V}) + w_{21}(\vec{V}\cdot\vec{V})\right] \end{aligned} \tag{4.75}$$

$$\widehat{F_0^{\rm eq}}(\vec{x},t) = z_0\,(\rho(\vec{x},t) - <\rho>) + \rho(\vec{x},t)\,z_{21}\,(\vec{V}\cdot\vec{V})$$

The new equilibrium population formulas are numerically better than the original ones because the term that used to be $w_0\,\rho$ is now $w_0\,(\rho - <\rho>)$. The new quantity $(\rho - <\rho>)$ is of the order $P/(3c^2 w_0)$ and the pressure P is of the order $\rho\,V^2$ as can

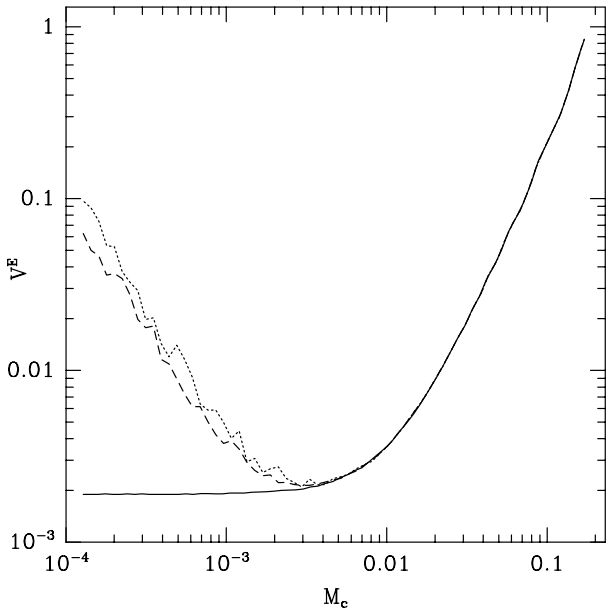

Figure 4-13: The error of the lattice Boltzmann method d2q7H is shown when single-precision arithmetic is used, when single-precision arithmetic together with the algebraic transformation of section 4.7.1 is used, and when double-precision arithmetic is used (dotted, dashed, solid lines).

be seen from the Navier Stokes equations. Hence the expression $w_0\,(\rho - <\rho>)$ is of the order $\rho(V/c)^2$. The new formulas compute the same quantities as the original formulas, and they incur a smaller loss of precision. Loss of precision still occurs when the terms proportional to (V/c) and $(V/c)^2$ are combined.

To verify the above analysis, figure 4-13 compares the error of the lattice Boltzmann method (d2q7H version) when single-precision arithmetic is used, when the algebraic transformation (together with single-precision arithmetic) is used, and when double-precision arithmetic is used (dotted, dashed, solid lines). The data comes from simulations of the hexagonal Taylor vortex with periodic boundary conditions and 30×30 grid. The error is plotted against M_c with Δt varying and is calculated at the final time $T = 1.0$. We see that when single-precision arithmetic is used, and the speed $\Delta x/\Delta t$ exceeds 300 (therefore $M_c < 0.003$), there is a growth of error that is caused by numerical roundoff. The algebraic transformation with single-precision arithmetic can reduce the roundoff error but can not prevent it. Double-precision arithmetic is

necessary to prevent the error growth in the Taylor vortex for $M_c < 0.003$.

4.7.2 Lattice gas methods

This section discusses some background material regarding the relation between lattice Boltzmann and lattice gas methods.

The lattice Boltzmann approach for simulating fluids is an outgrowth of the lattice gas approach [20, 15, 58, 23]. Both of these approaches have their origins in the kinetic theory of gases. The common idea behind them is that the advection and collision of particles can lead to the Navier-Stokes equations when the collision of particles conserves mass, momentum, and energy. Furthermore, the particles must move along the edges of a numerical grid that is highly symmetric [20, 15, 58] and is called a lattice. For example, typical grids in two dimensions are the hexagonal and the orthogonal lattices. In three dimensions, a cubic lattice is commonly used.

One difference between the lattice gas and the lattice Boltzmann approach is that the former represents the lattice particles with binary values 0 or 1, while the latter represents the particles with floating-point numbers. A binary value 0 or 1 represents the absence or presence of a single particle, while a floating-point number represents a density of particles. The change from single-bit variables to floating-point numbers has important consequences. From a mathematical point of view, the lattice Boltzmann method is easier to analyze and more flexible than the lattice gas method. In addition, the lattice Boltzmann method does not require averaging to remove statistical noise as does the lattice gas method.

One advantage of lattice gas over lattice Boltzmann is that single-bit operations may be desirable for special-purpose computers and for future technologies (quantum-bit computers have been mentioned in this context). Today, almost all computers are designed for floating-point operations, and they are well-suited for the lattice Boltzmann approach. However, special purpose computers have been built for single-bit operations of the lattice gas approach [53], and they are promising.

In comparing lattice gas and lattice Boltzmann approaches, it helps to view the lattice Boltzmann method as a lattice gas method with a very large number of particles per direction as opposed to one or zero particles. Further, we may observe that the number of velocity directions at each lattice node is a small number for the lattice Boltzmann method (to reduce computer memory requirements), while it varies from small to large for lattice gas methods. This is important because it has been reported [17] that lattice gas methods with a large number of velocity directions are more flexible and closer to correct hydrodynamics than lattice gas methods with a very small number of velocity directions. If this is true, then we may conclude that there are two ways to improve lattice gas methods: either by increasing the number of particles per direction (which eventually produces lattice Boltzmann), or by increasing the number of velocity directions per lattice node.

To carry the above discussion further, we may ask, "what about intermediate schemes which increase both the number of directions per node and the number of particles per direction?" For example, the 9-speed lattice Boltzmann model of section 4.3 uses 9 double-precision floating point numbers (64 bits $\times 9$) per lattice node because it has 8 directions and one non-moving population. An intermediate scheme with equivalent amount of memory might use 72 directions per lattice node with 2^8 particles (one byte) per direction. Would such an intermediate scheme perform better than lattice gas and lattice Boltzmann? In general, the question is to find the optimal distribution of bit-information to the physical degrees of freedom (number of directions, and number of particles per direction). This is an unsolved problem.

Chapter 5

Artificial-viscosity filter

This chapter discusses the need for an artificial-viscosity filter for dissipating numerical instabilities of high spatial frequency. Such a filter must be used both with the lattice Boltzmann method and with the compressible finite difference method of section 3.3 for flows with high Reynolds number.

Similar types of artificial-viscosity filters have been traditionally used in simulations of supersonic and transonic flow (Peyret&Taylor [38]). The idea of artificial-viscosity filters goes back to Richtmeyer&Morton [43] and perhaps earlier. However, a theoretical analysis of such filters is lacking, as far as I know. The analysis presented below is a first step towards a better understanding of artificial-viscosity filters.

5.1 Evidence of high-frequency oscillations

One of the difficulties of simulating subsonic compressible flow is the appearance of slow-growing high-frequency oscillations in the computed solution. These oscillations persist for a long time before they eventually overwhelm the solution and cause an exponential blow-up. The spatial wavelength of the oscillations is of the order of the mesh size Δx. The conditions that seem to trigger the oscillations include impulsive changes of density, high speed flow, and small viscosity, high Reynolds number flow.

Flow examples include the uniform flow past a sharp obstacle at high speed, and a jet of air impinging the labium of a flue pipe (see figures 5-1 and 5-1).

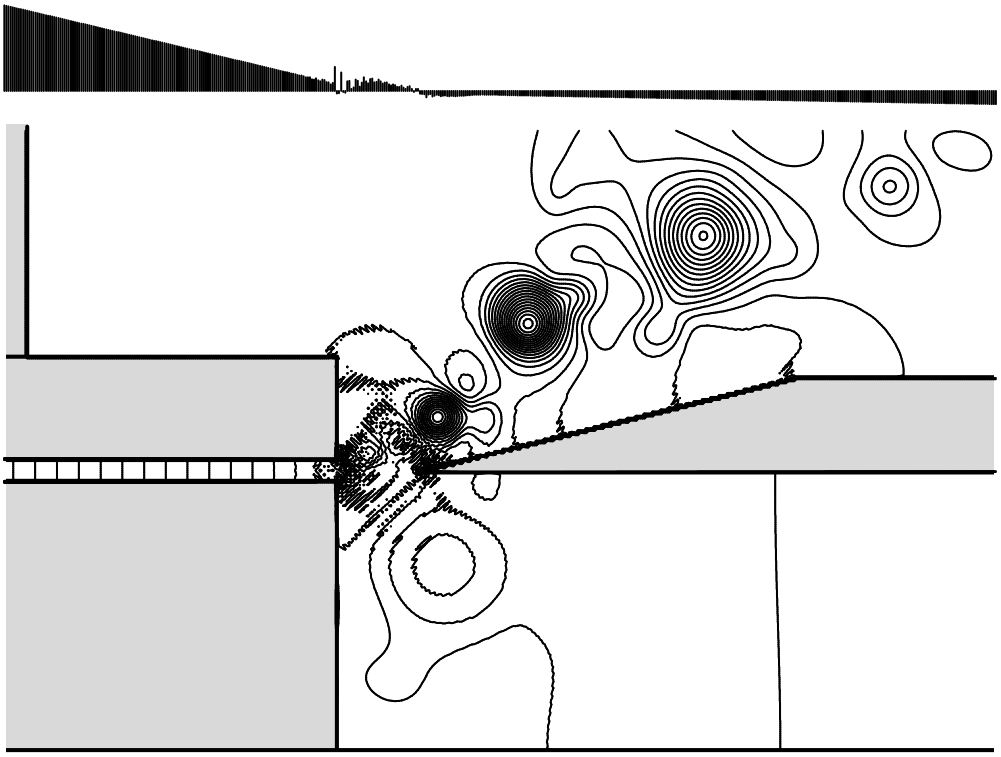

Figure 5-1: Iso-density contours in the flue-labium region, mean blowing velocity 1104 cm/s. High spatial frequencies cause instabilities if left un-treated.

Figures 5-1 and 5-1 show snapshots of the density in simulations which would become unstable without the use of an artificial-viscosity filter. In particular, the flue-labium region of a flue pipe is shown. The lattice Boltzmann method is used together with a fourth-order artificial-viscosity filter with $\alpha = 0.008$ (explained below). Iso-density contours are plotted, and also a horizontal cut of the density is shown at the top of the picture. The horizontal cut starts from the bottom surface of the flue channel, and continues parallel to and under the labium. High-frequency variations of density can be seen at the region between the flue and the labium in both simulations. Such high-frequency disturbances can cause instabilities if left untreated.

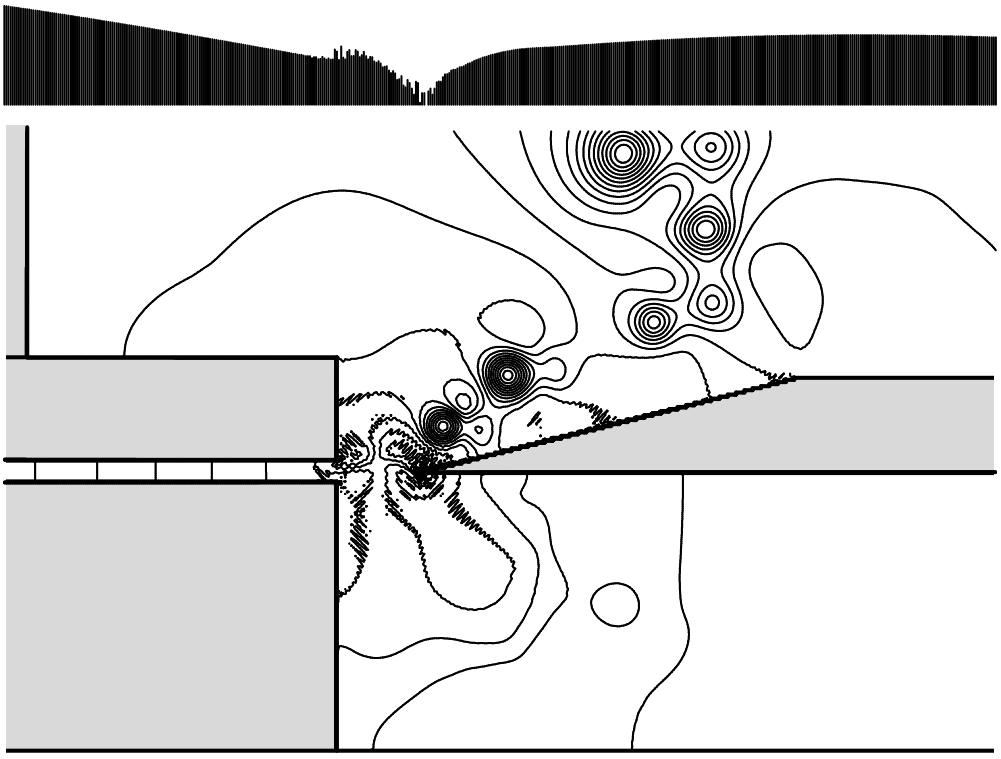

Figure 5-2: Iso-density contours in the flue-labium region, mean blowing velocity 1995 cm/s. High spatial frequencies cause instabilities if left un-treated.

5.2 The fourth-order filter

The high-frequency oscillatory instabilities can be mitigated by using a fourth-order artificial-viscosity filter as follows,

$$V^{n+1} = V^n + \alpha \left(\Delta x^4 \frac{\partial^4 V^n}{\partial x^4} + \Delta y^4 \frac{\partial^4 V^n}{\partial y^4} \right) \tag{5.1}$$

The above filter is applied at the end of every integration step to all three variables ρ, V_x, V_y. The parameter α controls the dissipation of the filter. In the case of the lattice Boltzmann method, a typical value of α is $\alpha = 0.008$. In the case of the compressible finite difference of section 3.3, a larger value α is used, typically $\alpha = 0.015$, because the finite difference method is more sensitive to instabilities than the lattice Boltzmann method. If α is too large, the solution is distorted (incorrect

physical modeling) and may even become unstable. According to a linear stability analysis which is described below, the largest value of α for stable 2D calculations is 1/16, namely $\alpha \leq 0.0625$. However, α should be less than 1/32 to produce a desirable filter (see figure 5-3). In practice, even smaller values of α are recommended (near 0.01) to avoid distorting the solution.

The discretization of the fourth-order filter in the x-direction is as follows,

$$\Delta^4 V_j \;=\; V_{j-2} - 4\,V_{j-1} + 6\,V_j - 4\,V_{j+1} + V_{j+2} \tag{5.2}$$

The above discretization is used at all the interior points which are at least 2 grid points away from the boundary. At the boundary points and at the next-to-boundary points, the above filter can not be applied for obvious reasons.

In order to filter the nodes near the boundary in a consistent way, a third-order differencing formula must be used at the next-to-boundary points as follows,

$$V_j^{n+1} = V_j^n \;+\; \alpha\,(V_{j-2} - 3\,V_{j-1} + 3\,V_j - \;V_{j+1}) \tag{5.3}$$

where the small index is $j \;=\; J - 1$ and the capital index J corresponds to the boundary point. Similar formulas must be used for the other boundary orientations. If the above formula is not used, stability problems may arise at the boundary.

A simple way to understand and to derive formula 5.3 is to consider the global conservation of the flow (total change in ρ, V_x, V_y) after the filter has been applied. To do so, the contributions of the filter must be summed at each grid point. For example, the total contribution of the filter at an interior point is zero: As the fourth-order stencil (equation 5.2) is shifted along the x-direction, an interior point V_j is multiplied by each one of the five "peaks" of the fourth-order stencil before being added-in, so that the total sum is zero. By contrast, the total contribution of the fourth-order stencil at points near the boundary (V_J to V_{J-3}) is generally non-zero. The third-order differencing formula adds-in the necessary corrections to make the total sum vanish, so that the filter obeys global conservation.

The third-order differencing formula at the next-to-boundary nodes plays an important role in the parallel-computing approach described in chapter 6. Normally, the boundaries of a simulation include the interior obstacles, and the perimeter that encloses the simulated region. In parallel simulations, additional boundaries arise because the global simulation region is divided into subregions which are computed in parallel. The crossing between two subregions is a kind of "artificial" boundary. Applying the fourth-order filter at the artificial boundary would require a lot of communication between the subregions (the fourth-order stencil requires two next neighbors). To save on communication, the fourth-order filter is *not applied* at the 'artificial" boundary. However, the third-order formula must be used, instead, for consistency. The author actually discovered the need for the third-order formula by noticing a slow-growing instability at the artificial boundary of a parallel simulation.

5.3 Analysis of fourth-order filter

The fourth-order filter can be understood by considering the dissipation of frequencies by a general m th-order filter,

$$V^{n+1} \;=\; V^n \;-\; \alpha\,\Delta x^m\,\frac{\partial^m V^n}{\partial x^m} \tag{5.4}$$

The analysis here treats the filter as an isolated system without considering the coupling between the filter and the numerical solution. We write V in terms of spatial frequencies κ,

$$\begin{aligned} V^n &= e^{i\kappa x} \\ V^{n+1} &= G e^{i\kappa x} \end{aligned} \tag{5.5}$$

where G is the growth factor, and the range of frequencies is $0 \le \kappa \le \pi/\Delta x$. By substituting equation 5.5 in equation 5.4, we obtain an estimate for the growth factor G of the m th-order filter,

$$G \;=\; 1 - i^m\,\alpha(\kappa\Delta x)^m \tag{5.6}$$

Here, the continuous version of the filter is considered for simplicity. The discretization of the filter is discussed below. We have the following cases,

$$\begin{array}{lll} m=2 & G=1+\alpha(\kappa\Delta x)^2 & \alpha \le 0 \\ m=3 & G=1+i\alpha(\kappa\Delta x)^3 & \text{unstable ?} \\ m=4 & G=1-\alpha(\kappa\Delta x)^4 & \alpha \ge 0 \\ m=6 & G=1+\alpha(\kappa\Delta x)^6 & \alpha \le 0 \\ m=8 & G=1-\alpha(\kappa\Delta x)^8 & \alpha \ge 0 \end{array} \tag{5.7}$$

The case $m = 2$ corresponds to physical viscosity, and therefore, it can *not* be used for artificial-viscosity filtering. The case $m = 3$ appears to amplify all frequencies for any choice of α, and furthermore the frequencies are phase-shifted disproportionately. Clearly, $m = 3$ is not a desirable filter, and similar conclusions hold for any odd integer m. The even integers m are suitable for filtering, and the smallest possible integer $m = 4$ corresponds to a fourth-order filter.

In comparing the even power filters, we may observe that the sign of α must alternate with increasing $m = 2, 4, 6, \ldots$ in order to produce a dissipative filter. Also, larger values of m produce "sharper" filters. A sharp filter means that the low frequencies are affected very little, and the high frequencies $\kappa\Delta x \sim \pi$ are strongly dissipated, and that the transition (cutoff) point is very abrupt. Finally, we may observe that the stability constraints on α become more stringent with increasing m. In particular, the condition $|G| < 1$ requires (for the continuous filter),

$$|\alpha| \;\le\; \frac{2}{(\kappa\Delta x)^m} \;\le\; \frac{2}{(\pi)^m} \tag{5.8}$$

The fourth-order filter $m = 4$ is a good choice because it has the desirable filtering behavior as shown below in more detail, and also because $m = 4$ is the smallest possible integer. The size of m is proportional to the computational cost of the filter, assuming that the filter is implemented via finite differences.

The discretization of the fourth-order filter based on symmetric differences is given

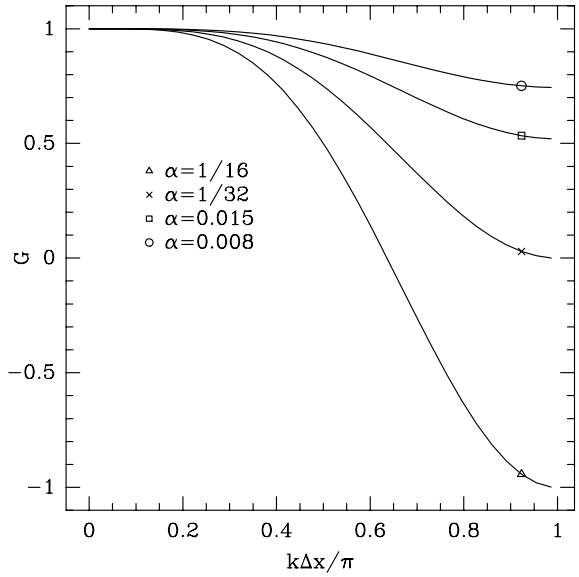

Figure 5-3: Amplification of spatial frequencies by the fourth-order artificial-viscosity filter (2D discretized) for different values of α.

by equation 5.2, and it produces the following growth factor,

$$\begin{aligned} G &= 1 - \alpha\,(6 - 4\,e^{i\kappa\Delta x} - 4\,e^{-i\kappa\Delta x} + e^{i2\kappa\Delta x} + e^{-i2\kappa\Delta x}) \\ &= 1 - \alpha\,(6 - 8\cos\kappa\Delta x + 2\cos 2\kappa\Delta x) \\ &= 1 - 4\,\alpha\,(1 - \cos\kappa\Delta x)^2 \end{aligned} \tag{5.9}$$

For stability purposes, the magnitude of the growth factor must be less than one,

$$-1 \le G \le 1 \tag{5.10}$$

Using the largest possible frequency $\kappa\Delta x = \pi$ we obtain,

$$0 \le \alpha \le \frac{1}{8} \tag{5.11}$$

In two dimensions, it is easy to see that the growth factor becomes,

$$G = 1 - 4\,\alpha\,\left[(1 - \cos\kappa_1\Delta x)^2 + (1 - \cos\kappa_2\Delta y)^2\right] \tag{5.12}$$

which implies the following limits on α,

$$0 \le \alpha \le \frac{1}{16} \tag{5.13}$$

The growth factor of equation 5.12 is plotted in figure 5-3 for different values of α. We can see that the maximum value for stability $\alpha = 1/16$ produces an undesirable filter because the very high frequencies are simply multiplied by a minus sign and are not dissipated. For a desirable filter, α should be

$$\alpha \leq \frac{1}{32} \tag{5.14}$$

In practice, even smaller values of α are preferred in order to prevent distortion of the solution. For example, the value $\alpha = 0.008$ produces very small dissipation of very high frequencies only. This small dissipation is needed in order to avoid the high-frequency numerical oscillations which appear in simulations of subsonic compressible flow.

5.4 Other kinds of filters

The frequency analysis presented above can be continued in order to understand further the artificial-viscosity filters. To this end, the shift operators S_{-1} and S_{+1} are introduced, and they look as follows in the frequency domain,

$$\begin{aligned} S_{-1}\, V &= e^{-i\kappa\Delta x}\, V \\ S_{+1}\, V &= e^{i\kappa\Delta x}\, V \end{aligned} \tag{5.15}$$

A second-order symmetric differencing formula can be written as follows,

$$(\Delta x)^2\, \delta_{xx}\, V \;=\; (S_{-1} - 2 + S_{+1})\, V \;=\; -2\,(1 - \cos\kappa\Delta x)\, V \tag{5.16}$$

The discretization of an m th-order filter for even $m = 2l$ based on symmetric differences can be found by applying l-times the above second-order difference operator,

$$V^{n+1} \;=\; V^n \;-\; \alpha\,(\Delta x)^{2l}\,(\delta_{xx})^l\, V^n \tag{5.17}$$

The growth factor is as follows (for a one-dimensional filter),

$$G \;=\; 1 \;-\; \alpha\,(-2)^l\,(1 - \cos\kappa\Delta x)^l \tag{5.18}$$

The above expression is a generalization of the fourth-order formula obtained previously for $m = 4$ or $l = 2$.

The frequency analysis can also be applied to "tophat" averaging filters in the context of fluid flow simulations. When high frequencies must be removied, tophat averaging is the first idea that comes to mind. For example, a two-point averaging formula is as follows,

$$V^{n+1} = \frac{(S_{-1} + S_{+1})}{2} V^n \tag{5.19}$$

The above filter is undesirable because it causes significant dissipation of low frequencies as well as high frequencies, and also because it causes phase distortion as can be seen from the imaginary part of the growth factor $(1 + e^{i\kappa\Delta x})/2$. Thus, we may try a three-point averaging formula,

$$V^{n+1} = \frac{(S_{-1} + 1 + S_{+1})}{3} V^n \tag{5.20}$$

The growth factor is $(1 + 2\cos\kappa\Delta x)/3$, and has no imaginary components which is good. However, the high frequencies $\kappa\Delta x \geq \pi/3$ are multiplied by a minus sign and are not dissipated completely. For example, the highest frequency $\kappa\Delta x = \pi$ is multiplied by $-1/3$. The 3-point averaging filter can be improved by considering a weighted 3-point averaging,

$$V^{n+1} = V^n (1 - \alpha) + \alpha \frac{(S_{-1} + 1 + S_{+1})}{3} V^n \tag{5.21}$$

The above expression is actually equivalent to a second-order viscosity filter as can be seen by rewriting it as follows,

$$V^{n+1} = V^n + \alpha \frac{(S_{-1} - 2 + S_{+1})}{3} V^n \tag{5.22}$$

Clearly, a weighted 3-point averaging filter is undesirable because it affects the physical viscosity. Furthermore, it is easy to see that the 4-point, 6-point, 8-point, etc averaging filters produce undesirable phase distortion. [1] Therefore, the smallest viable

[1] A discussion of phase distortion according to an Electrical Engineering textbook can be found in Siebert [47, p.472].

choice is a 5-point averaging filter. The general form of a weighted 5-point averaging filter, which does not cause phase distortion, can be written as follows where β, γ are real-numbers (weighting factors),

$$V^{n+1} \;=\; V^n\,(1-\alpha) \;+\; \alpha\,\frac{\beta S_{-2} + \gamma S_{-1} + 1 + \beta S_{+1} + \gamma S_{+2}}{1 + 2\beta + 2\gamma}\,V^n \tag{5.23}$$

The fourth-order artificial-viscosity filter of equation 5.2 is a special case of the above expression. This analysis puts in perspective the fourth-order artificial-viscosity filter, and shows why the 2-point, 3-point, and 4-point averaging filters do not perform well in fluid flow simulations, a fact which can be easily tested in actual simulations.

5.5 The origin of high-frequency oscillations

The origin of the slow-growing high-frequency numerical oscillations in simulations of compressible flow is not well understood. It is possible that the triggering of the oscillations is both numerical and physical. Peyret&Taylor [38, p.323] report that high-frequency oscillations appear both in explicit and implicit methods for transonic and supersonic compressible flow, which hints that there may be a physical cause that triggers the oscillations.

It has been conjectured (Fletcher [18, p.438] and elsewhere) that physical turbulence may be triggering the numerical oscillations. Turbulent flow produces high frequency disturbances whose wavelength is much smaller than the limited resolution of computer simulations. Accordingly, it has been conjectured that a type of frequency aliasing may be happening from the turbulent length scales to the coarser length scales of the simulation. However, the details of such a mechanism have never been shown, and they are not obvious. In particular, the algebraic system of difference equations (the simulation) is *not a sampling process* of the underlying differential equations of fluid flow. Perhaps, a more plausible conjecture is that the discrete system of equations inherits a tendency for a kind of "discrete turbulence" from the continuous equations of fluid flow.

A related point is that physical turbulence provides a mechanism for dissipating very high-frequency oscillations. This is the energy cascade idea: the energy of the flow cascades from large scale motion to smaller and smaller vortices until being dissipated. Perhaps, the turbulent dissipation can be compared with the fourth-order artificial-viscosity dissipation. This idea is the reason why fourth-order artificial viscosity is sometimes referred to as a model of subgrid turbulence. However, a lot of work remains to be done to understand how good (or how bad) a model of subgrid turbulence is the fourth-order artificial viscosity.

This chapter completes the basic discussion of numerical methods and numerical modeling. In the next chapter, the parallel computation of fluid dynamics is discussed. Subsequently, in chapter 7 examples of simulations of flue pipes are presented which complement the simulations already presented in chapter 1.

Chapter 6

Parallel Computing

6.1 Introduction

This chapter presents an effective approach of simulating fluid dynamics on a cluster of non-dedicated workstations. Concurrency is achieved by decomposing the flow problem into subregions, and by assigning the subregions to parallel subprocesses. The use of explicit numerical methods leads to small communication requirements. The parallel subprocesses automatically migrate from busy hosts to free hosts in order to exploit the unused cycles of non-dedicated workstations, and to avoid disturbing the regular users. The system is straightforwardly implemented on top of UNIX and TCP/IP communication routines.

Typical simulations achieve 80% parallel efficiency (speedup/processors) using 20 HP-Apollo workstations in a cluster where there are 25 non-dedicated workstations total. Detailed measurements of efficiency in simulating two and three-dimensional flows are presented, and a theoretical model of efficiency is developed which fits closely the measurements. Two numerical methods of fluid dynamics are tested: finite differences and the lattice Boltzmann method. Further, it is shown that the shared-bus Ethernet network is adequate for two-dimensional simulations of fluid dynamics, but limited for three-dimensional ones. It is expected that new technologies in the

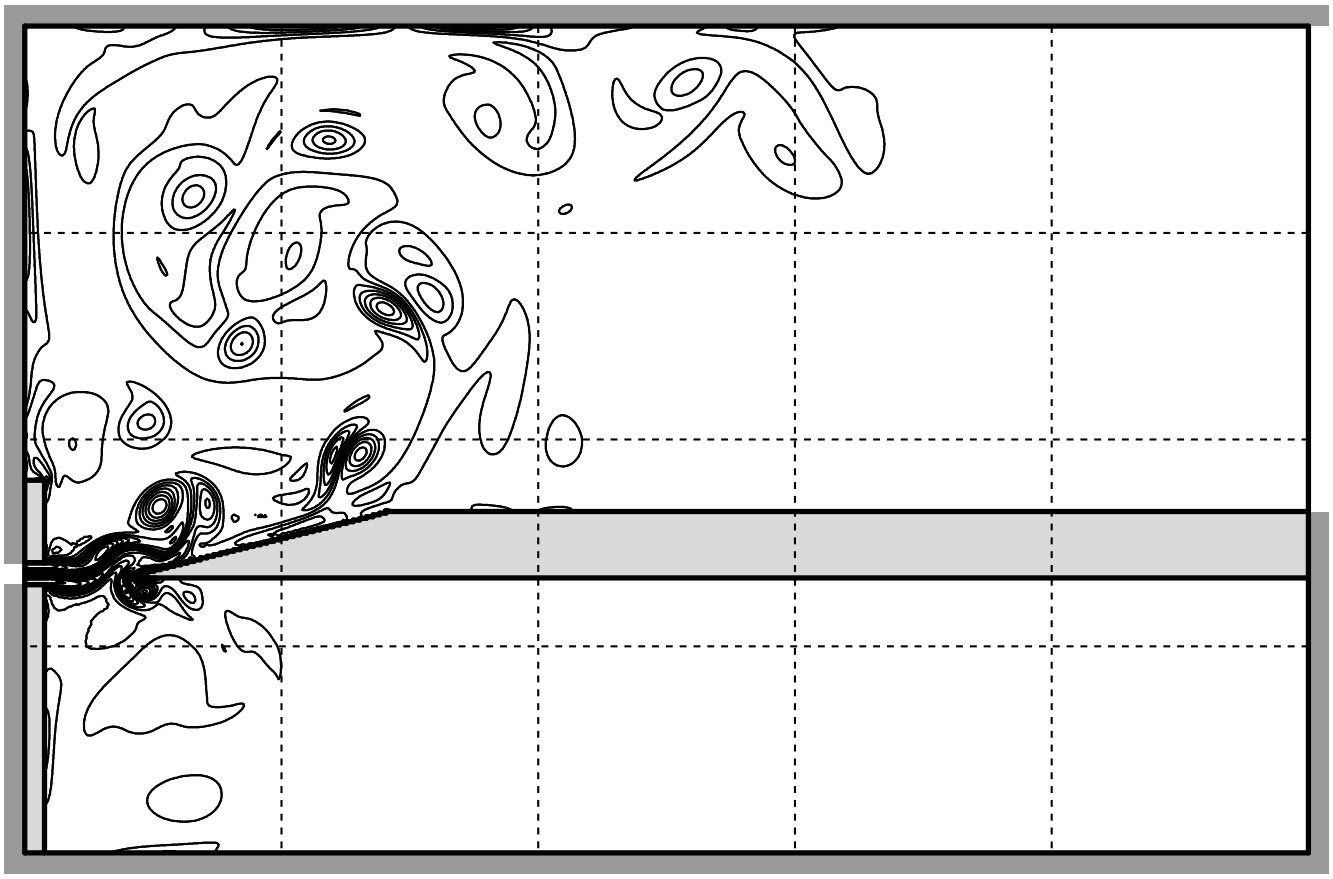

Figure 6-1: Simulation of flue pipe using 20 workstations in 5×4 decomposition.

near future such as Ethernet switches, FDDI and ATM networks will make practical three-dimensional simulations of fluid dynamics on a cluster of workstations.

The parallel system presented here is well-suited for simulating subsonic flows which involve both hydrodynamics and acoustic waves; for example, the flow of air inside wind musical instruments. Such flow problems favor the use of explicit methods (see section 3.2) which are perfectly parallelizable, and lead to low communication requirements between parallel processes. The use of explicit methods is important for parallel computing on a cluster of workstations because the communication capacity between workstations is usually small.

In general, the use of explicit methods is recommended in situations where increasing numbers of local processing units are available with minimum communica-

tion capacity between the processing units. Such computers may be widespread in the future; for instance, a future parallel computer may consist of millions of local processing units, each unit having the power of one of today's workstations. With this perspective in mind, the work presented herein for a cluster of 20 to 25 workstations, may have applications for future parallel computers as well.

Outline

Section 6.2 presents some examples of parallel simulations which demonstrate the power of the present approach, and also help to motivate the subsequent sections. Section 6.3 reviews parallel computing and local-interaction problems in general. Sections 6.4 and 6.5 describe the implementation of the parallel simulation system, including the automatic migration of processes from busy hosts to free hosts. Section 6.6 explains the parallelization of numerical methods for fluid dynamics. Finally, sections 6.7 and 6.8 measure experimentally the performance of the parallel system, and also develop a theoretical model of parallel efficiency for local-interaction problems which fits well the measured efficiency.

Most issues are discussed as generally as possible within the context of local-interaction problems, and the specifics of fluid dynamics are limited to section 6.2 and section 6.6.

6.2 Examples of distributed simulations

The parallel simulation system is used to simulate subsonic flow, and in particular, the flow of air inside flue pipes of wind musical instruments such as the organ, the recorder, and the flute. This is a phenomenon that involves the interaction between hydrodynamic flow and acoustic waves: When a jet of air impinges a sharp obstacle in the vicinity of a resonant cavity, the jet begins to oscillate strongly, and it produces audible musical tones. The jet oscillations are reenforced by a nonlinear feedback

from the acoustic waves to the jet. Similar phenomena occur in human whistling and in voicing of fricative consonants (Shadle [46]). Although sound-producing jets have been studied for more than a hundred years, they remain the subject of active research (Verge94 [57, 56], Hirschberg [26]) because they are very complex.

The parallel system presented herein can easily simulate flue pipes using uniform orthogonal grids as large as 1200×1200 in two dimensions (1.5 million nodes) and even larger. Typically, smaller grids are employed, however, such as 800×500 (0.38 million nodes) in order to reduce the computing time. For example, if we divide a 800×500 grid into twenty subregions and assign each subregion to a different HP9000/700 workstation, we can compute 70,000 integration steps in 12 hours of run time. This produces about 12 milliseconds of simulated time, which is long enough to observe the initial response of a flue pipe with a jet of air that oscillates at 1000 cycles per second.

Figure 6-1 shows a snapshot of a 800×500 simulation of a flue pipe by plotting equi-vorticity contours (the curl of fluid velocity). The decomposition of the two-dimensional space $(5 \times 4) = 20$ is shown as dashed lines superimposed on top of the physical region. The gray areas are walls, and the dark-gray areas are walls that enclose the simulated region and demarcate the inlet and the outlet. The jet of air enters from an opening on the left wall, impinges the sharp edge in front of it, and it eventually exits from the simulation through the opening on the right part of the picture. The resonant pipe is located at the bottom part of the picture.

Figure 6-2 shows a snapshot of another simulation that uses a slightly different geometry than figure 6-1. In particular, figure 6-2 includes a long channel through which the jet of air must pass before impinging the sharp edge. Also, the outlet of the simulation is located at the top of the picture as opposed to the right. This is convenient because the air tends to move upwards after impinging the sharp edge. Overall, figure 6-2 is a more realistic model of flue pipes than figure 6-1.

From a computational point of view the geometry of figure 6-2 is interesting be-

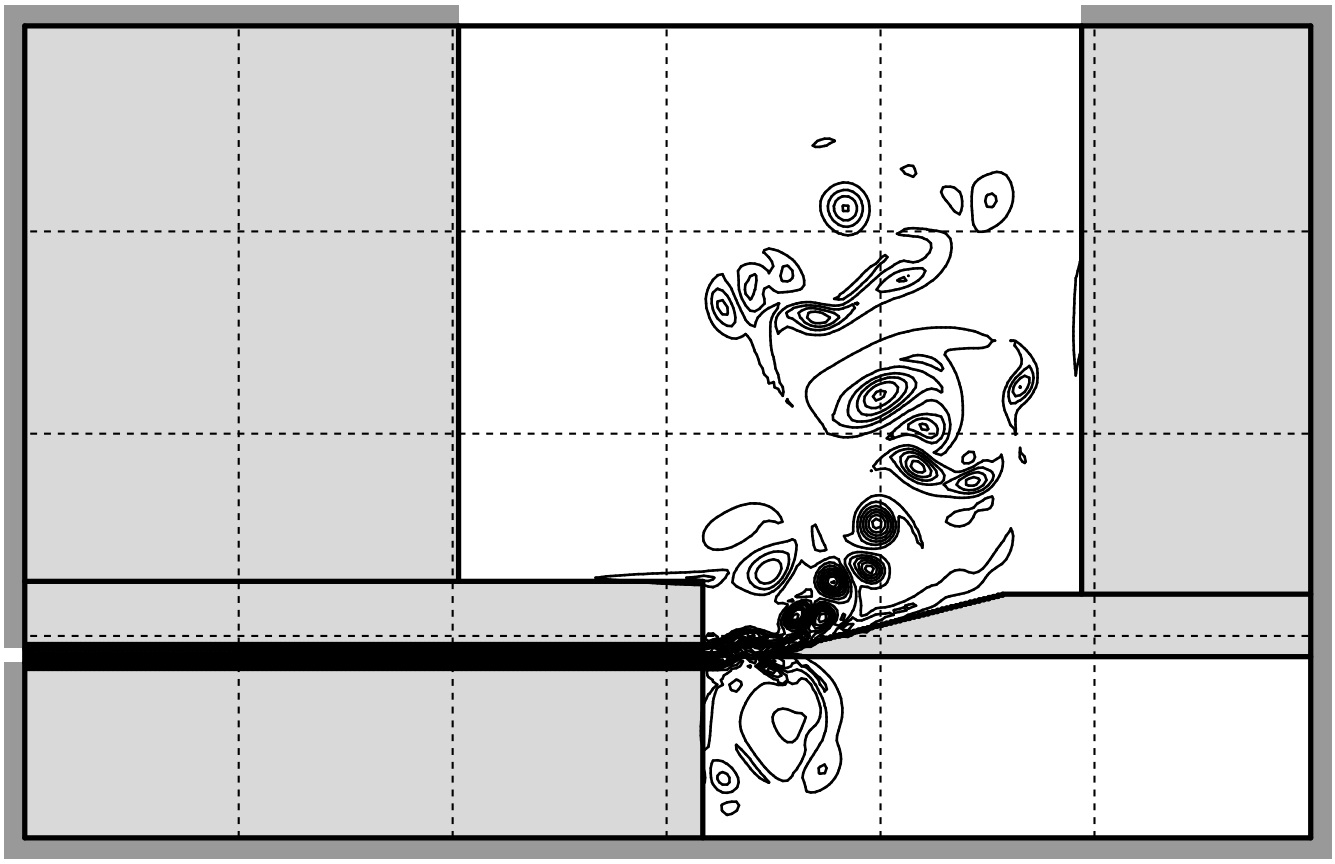

Figure 6-2: Simulation of a flue pipe using 15 workstations in 6×4 decomposition with 9 subregions inactive.

cause there are subregions that are entirely gray, i.e. they are entirely solid walls. Consequently, these subregions need not be assigned to any workstation. Thus, although the decomposition is $(6 \times 4) = 24$, only 15 workstations are employed for this problem. In terms of the number of grid nodes, the full rectangular grid is 1107×700 or 0.7 million nodes, but only 15/24 of the total nodes or 0.48 million nodes are simulated. This example shows that an appropriate decomposition of the problem can reduce the computational effort in some cases, as well as provide opportunities for parallelism. More sophisticated decompositions can be even more economical than the present ones. Uniform decompositions and identical-shaped subregions are employed here because they are very simple.

The above simulations have been performed using the lattice Boltzmann method.

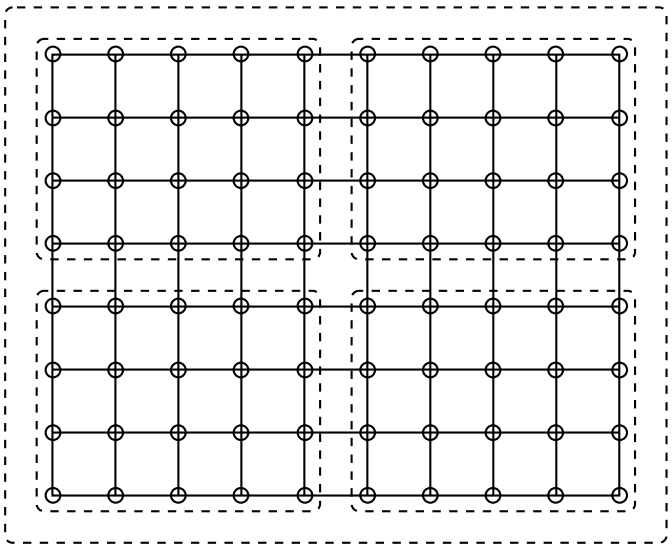

Figure 6-3: A problem of local interactions in two dimensions, and its decomposition (2×2) into four subregions.

Similar results are obtained using a finite difference approach. Further issues on parallelization of fluid dynamics are discussed in section 6.6. Next, the basics of local-interaction problems are reviewed, and the implementation of the parallel system is described. These issues are important for understanding in detail how the parallel system works and why it works well.

6.3 Local-interaction computations

We define a local-interaction computation as a set of "parallel nodes" that can be positioned in space so that the nodes interact only with neighboring nodes. For example, figure 6-3 shows a two-dimensional space of parallel nodes which are connected by solid lines representing the local interactions. In this example, the interactions extend to a distance of one neighbor, and have the shape of a star stencil, but other patterns of local interactions are also possible. Figure 6-4 shows two typical interactions which extend to a distance of one neighbor, a star stencil and a full stencil.

The parallel nodes of a local-interaction problem are the finest grain of parallelism

that is available in the problem; namely, they are the finest decomposition of the problem into units that can evolve in parallel after communication of information with their neighbors. In practice, the parallel nodes are usually grouped into subregions of nodes, as shown in figure 6-3 by the dashed lines. Each subregion is assigned to a different processor, and the problem is solved in parallel by executing the following sequence of steps repeatedly,

- Calculate the new state of the interior of the subregion using the previous history of the interior as well as the current boundary information from the neighboring subregions.

- Communicate boundary information with the neighboring subregions in order to prepare for the next local calculation.

The boundary that is communicated between subregions is the outer surface of the subregions. Section 6.4.2 describes a good way of organizing this communication.

Local-interaction problems are highly-suited for parallel computing because the communication is local, and also because the amount of communication relative to computation can be controlled by varying the decomposition. In particular, when each subregion is as small as one node (one processor per node), there is maximum parallelism, and a lot of communication relative to the computation of each processor. As the size of each subregion increases (which is called "coarse-graining"), both the parallelism and the the amount of communication relative to computation decrease. This is because only the surface of a subregion communicates with other subregions. Eventually, when one subregion includes all the nodes in the problem, there is no parallelism and no need for communication anymore. Somewhere between these extremes, we often find a good match between the size of the subregion (the "parallel grain size") and the communication capabilities of the computing system. This is the reason why local-interaction problems are very flexible and highly desirable for parallel computing.

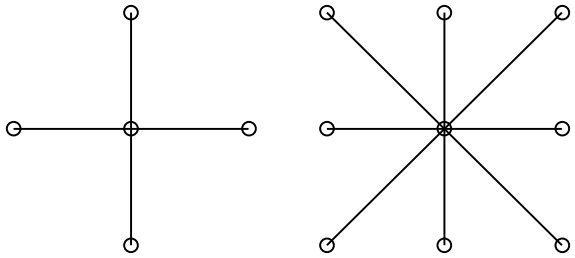

Figure 6-4: A star stencil and a full stencil represent two typical nearest neighbor local interactions.

6.4 The distributed system

The design of the parallel system follows the basic ideas of local-interaction parallel computing that are discussed above. This section describes the implementation of the parallel system, which is based on UNIX and TCP/IP communication routines, and exploits the common file system of the workstations.

6.4.1 The main modules

For the sake of programming modularity, the parallel simulation system is organized into the following four modules:

- The initialization program produces the initial state of the problem to be solved as if there was only one workstation.

- The decomposition program decomposes the initial state into subregions, generates local states for each subregion, and saves them in separate files, called "dump files". These files contain all the information that is needed by a workstation to participate in a distributed computation.

- The job-submit program finds free workstations in the cluster, and begins a parallel subprocess on each workstation. It provides each process with a dump

file that specifies one subregion of the problem. The processes execute the same program on different data.

- The monitoring program runs every few minutes and checks that the parallel processes are progressing correctly. If an unrecoverable error occurs, the distributed simulation is stopped, and a new simulation is started from the last state which is saved automatically every $10 - 20$ minutes. If a workstation becomes too busy, automatic migration of the affected process takes place, as explained in section 6.5.

All of the above programs (initialization, decomposition, submit, and monitoring) are performed by one designated workstation in the cluster. Although it is possible to perform the initialization and the decomposition in a distributed fashion in principle, a serial approach is chosen here for simplicity.

Regarding the selection of free workstations, the strategy is to separate all the workstations into two groups: workstations with active users, and workstations with idle users (meaning more than 20 minutes idle time). An idle-user does not necessarily imply an idle workstation because background jobs may be running; however, an idle-user is preferred to an active user. Thus, the idle-user workstations are examined first to see if the fifteen-minute average of the CPU load is below a pre-set value, in which case the workstation is selected. For example, the load must be less than 0.6 where 1.0 means that a full-time process is running on the workstation. After examining the idle-user workstations, the active-user workstations are examined, and the search continues as long as more workstations are needed.

In addition to the above programs (initialization, decomposition, submit, and monitoring), there is also the program that is executed in parallel by all the workstations. This program consists of two steps: "compute locally", and "communicate with neighbors". Below we discuss issues relating to communication.

6.4.2 Communication

The communication between parallel processes synchronizes the processes in an indirect fashion because it encourages the processes to begin each computational cycle together with their neighbors as soon as they receive data from their neighbors. Thus, there is a local near-synchronization which also encourages a global near-synchronization. However, neither local nor global synchronization is guaranteed, and in special circumstances the parallel processes can be several integration time steps apart. This is important when a process migrates from a busy host to a free host, as explained in section 6.5 (also see the appendix).

The communication of data between processes is organized by means of a well-known programming technique which is called "padding" or "ghost cells" (Fox [19], Camp [6]). Specifically, each subregion is padded with one or more layers of extra nodes on the outside. One layer of nodes is used if the local interaction extends to a distance of one neighbor, and more layers are used if the local interaction extends further. Once the data is copied from one subregion onto the padded area of a neighboring subregion, the boundary values are available locally during the current cycle of the computation. This is a good way to organize the communication of boundary values between neighboring subregions.

In addition, padding leads to programming modularity in the sense that the computation does not need to know anything about the communication of the boundary. As long as we compute within the interior of each subregion, the computation can proceed as if there was no communication at all. Because of this separation between computation and communication, it is possible to develop a parallel program as a straightforward extension of a serial program. In the present system, the fluid dynamics code can be compiled either into a parallel program or into a serial program depending on the settings of a few C-compiler directives. The main differences between the parallel and the serial programs are the padded areas, and a subroutine that communicates the padded areas between processes.

The subroutine that communicates the padded areas between processes is implemented using "sockets" and the TCP/IP protocol. A socket is an abstraction in the UNIX operating system that provides system calls to send and receive data between UNIX processes on different workstations. A number of different protocols (types of behavior) are available with sockets, and TCP/IP is the simplest one. This is because the TCP/IP protocol guarantees delivery of any messages sent between two processes. Accordingly, the TCP/IP protocol behaves as if there are two first-in-first-out channels for writing data in each direction between two processes. Also, once a TCP/IP channel is opened at startup, it remains open throughout the computation except during migration when it must be re-opened, as explained later.

Opening the TCP/IP channel involves a simple hand-shaking, "I am listening at this port number. I want to talk to you at this port number? Okay, the channel is open." The port numbers are needed to identify uniquely the sender and the recipient of a message so that messages do not get mixed up between different UNIX processes. Further, the port numbers must be known in advance before the TCP/IP channel is opened. Thus, each process must first allocate its port numbers for listening to its neighbors, and then write the port numbers into a shared file. The neighbors must read the shared file before they can connect using TCP/IP.

6.5 Transparency to other users

The basic operation of the parallel simulation system was described in the previous section. Here, the issues that arise when sharing the workstations with other users are discussed. Specifically, there are two issues to consider: sharing the CPU cycles of each workstation, and sharing the local-area network and the file server. The sharing of CPU cycles is achieved by employing an automatic migration of processes from busy hosts to free hosts as explained below.

6.5.1 Automatic migration of processes

The utilization of a workstation can be distinguished into three basic categories:

- (i) The workstation is idle.
- (ii) The workstation is running an interactive program that requires fast CPU response and few CPU cycles.
- (iii) The workstation is running another full-time process in addition to a parallel subprocess.

In the first two cases, it is appropriate to time-share the workstation with another user. Furthermore, it is possible to make the distributed computation transparent to the regular user of the workstation by assigning a low runtime priority to the parallel processes (UNIX command "nice"). Because the regular user's tasks run at normal priority, they receive the full attention of the processor immediately, and there is no loss of interactiveness. After the user's tasks are serviced, there are enough CPU cycles left for the distributed computation.

In the third case, when a workstation is running another full-time process in addition to a parallel subprocess, the parallel process must migrate to a new host that is free. This is because the parallel process interferes with the regular user, and further, the whole distributed computation slows down because of the busy workstation. Clearly, such a situation must be avoided.

The parallel system detects the need for migration using the monitoring program described in the previous section. The monitoring program checks the CPU load of every workstation via the UNIX command "uptime", and signals a request for migration if the five-minute-average load exceeds a pre-set value, typically 1.5. The intent is to migrate only if a second full-time process is running on the same host, and to avoid migrating too often. In the present system, there is typically one migration every 45 minutes for a distributed computation that uses 20 workstations from a pool

of 25 workstations. Also, each migration lasts about 30 seconds. Thus, the cost of migration is insignificant because the migrations do not happen too often.

During a migration, a precise sequence of events takes place in order for the migration to complete successfully,

- The affected process A receives a signal to migrate.
- All the processes get synchronized.
- Process A saves its state into a dump file, and stops running.
- Process A is restarted on a free host, and the distributed computation continues.

Signals for migration are sent through an interrupt mechanism, "kill -USR2" (see UNIX manual). In this way, both the regular user of a workstation and the monitoring program can request a parallel subprocess to migrate at any time.

The reason for synchronizing all the processes prior to migration, is to simplify the restarting of the processes after the migration has completed. In addition, the synchronization allows more than one process to migrate at the same time if it is desired. A synchronization scheme is employed which instructs all the processes to continue running until a chosen synchronization time step, and then to pause for the migration to take place. The details of the synchronization scheme are described in the appendix.

When all the processes reach the synchronization time step, the processes that need to migrate save their state and exit, while they notify the monitoring program to select free workstations for them. The other parallel processes suspend execution and close their TCP/IP communication channels. When the monitoring program finds free hosts for all the migrating processes, it sends a *CONT* signal to the waiting processes. In response, all the processes re-open their communication channels, and the distributed computation continues normally.

Overall, the migration mechanism is designed to be as simple as possible. In fact, it is equivalent to stopping the computation, saving the entire state on disk, and then

restarting; except, only the state of the migrating process is saved on disk. In contrast to this simple migration mechanism, the migration of processes is a challenging task in a general computing environment such as a distributed operating system [16]. In the present system, the migration task has been simplified because the parallel processes have been designed appropriately to accommodate migration easily.

6.5.2 Sharing the network and file server

A related issue to sharing the workstations with other users, is the sharing of the network and the file server. A distributed program must be carefully designed to make sure that the system does not monopolize the network and the file server. Abuse of shared resources is very easy in today's UNIX operating system because there are no direct mechanisms for controlling or limiting the use of shared resources. Thus, a program such as FTP (file transfer) is free to send many megabytes of data through the network, and to monopolize the network, so that the network appears "frozen" to other users. A distributed program can monopolize the network in a similar way, if it is not designed carefully.

The present parallel distributed system does not monopolize the network because it includes a time delay between successive send-operations, during which the parallel processes are calculating locally. Moreover, the time delay increases with the network traffic because the parallel processes must wait to receive data before they can start the next integration step. Thus, there is an automatic feedback mechanism that slows down the distributed computation, and allows other users to access the network at the same time.

Another situation to consider is when the parallel processes are writing data to the common file system. Specifically, when all the parallel processes save their state on disk at approximately the same time (a couple of megabytes per process), it is very easy to saturate both the network and the file server. In order to avoid this situation, a constraint is imposed that the parallel processes must save their state

one after the other in an orderly fashion, allowing sufficient time gaps between, so that other programs can use the network and the file system. Thus, a saving operation that would take 30 seconds and monopolize the shared resources, now takes $60 - 90$ seconds but leaves free time slots for other programs to access the shared resources at the same time. Overall, a careful design has made the distributed system mostly transparent to the regular users of the workstations.

6.6 Fluid dynamics

Having described the basic operation of the distributed system, I now discuss the parallelization of two numerical methods for simulating fluid dynamics: the explicit finite difference method, and the lattice Boltzmann method. Both of these methods are explicit, and are well-suited for simulating subsonic flow which involves both hydrodynamics and acoustic waves. Further, both methods are well-suited for parallel computing because they employ local interactions.

The explicit finite difference method is described in detail in chapter 3, and is a straightforward discretization of the Navier Stokes equations. Specifically, the spatial derivatives are discretized using centered differences on a uniform orthogonal grid, and the time derivatives are discretized using forward Euler differences. For the purpose of improving numerical stability, the density equation is updated using the values of velocity at time $t + \Delta t$. In other words, the velocities values are computed first, and then the density values are computed as a separate step. The precise sequence of computational steps for the finite difference method is as follows,

- Calculate V_x, V_y (inner)
- Communicate: send/recv V_x, V_y (boundary)
- Calculate ρ (inner)
- Communicate: send/recv ρ (boundary)

- Filter ρ, V_x, V_y (inner)

The filter that is included above is crucial for simulating subsonic flow at high Reynolds number (fast moving flow). The simulation of subsonic compressible flow is susceptible to slow-growing numerical oscillations. The filter prevents instabilities by dissipating high spatial frequencies whose wavelength is comparable to the grid mesh size (the distance between neighboring fluid nodes). The same filter is used both with the finite difference method and with the lattice Boltzmann method. A detailed description of the filter can be found in chapter 5.

We recall from chapter 4 that the lattice Boltzmann method uses two kinds of variables to represent the fluid, the traditional fluid variables ρ, V_x, V_y, and another set of variables called populations F_i. During each cycle of the computation, the fluid variables ρ, V_x, V_y are computed from the F_i, and then the ρ, V_x, V_y are used to relax the F_i. Subsequently, the relaxed populations are shifted to the nearest neighbors of each fluid node, and the cycle repeats. The precise sequence of computational steps for the lattice Boltzmann method is as follows,

- Relax F_i (inner)
- Shift F_i (inner)
- Communicate: send/recv F_i (boundary)
- Calculate ρ, V_x, V_y from F_i (inner)
- Filter ρ, V_x, V_y (inner)

Regarding the communication of boundary values by the finite difference method (FD) and the lattice Boltzmann method (LB), there are some differences that will become important in the next two sections, when the performance of the parallel simulation system is examined. The first difference is that FD sends two messages per computational cycle as opposed to LB which sends all the boundary data in

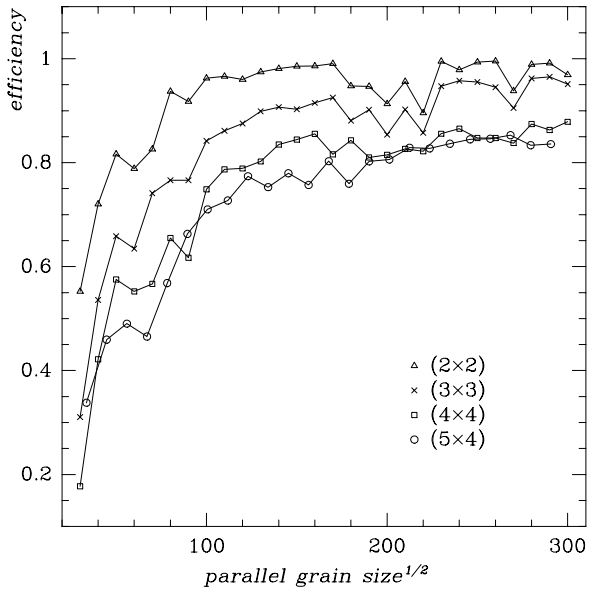

Figure 6-5: Parallel efficiency in 2D simulations using lattice Boltzmann.

one message. This results in slower communication for FD when the messages are small because each message has a significant overhead in a local-area network. The second difference is that LB communicates 5 variables (double precision floating-point numbers) per fluid node in three dimensional problems, while FD communicates only 4 variables per fluid node. In two dimensional problems, both methods communicate 3 variables per fluid node.

6.7 Experimental measurements of performance

The performance of the parallel simulation system has been measured when using the finite difference method and the lattice Boltzmann method to simulate a well-known problem in fluid mechanics, Hagen-Poiseuille flow through a rectangular channel (Skordos [48] and Landau&Lifshitz [32, p.51]). Below, measurements of the parallel efficiency f and the speedup S are presented. These numbers are defined as follows,

$$f = \frac{S}{P} = \frac{T_1}{P\,T_p} \tag{6.1}$$

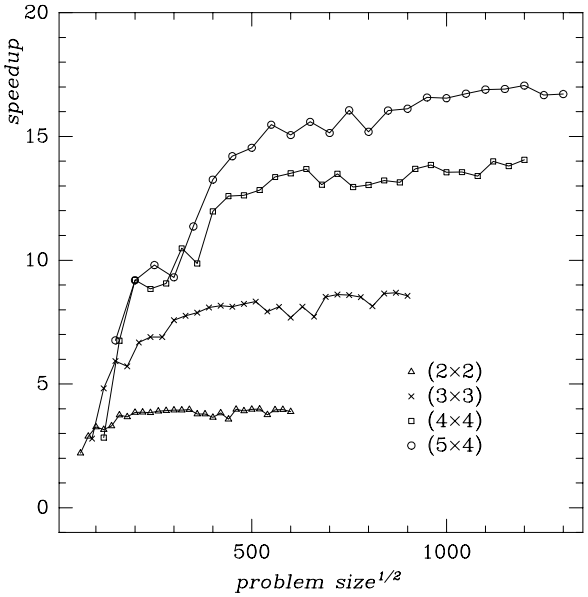

Figure 6-6: Parallel speedup in 2D simulations using lattice Boltzmann.

where T_p is the elapsed time for integrating a problem using P processors, and T_1 is the elapsed time for integrating the same problem using a single processor. The times T_p and T_1 for integrating a problem are measured by averaging over 20 consecutive integration steps, and also by averaging over each processor that participates in the parallel computation. The resulting average is the time interval it takes to perform one integration step. The UNIX system call "gettimeofday" is used to obtain accurate timings. Although most measurements are taken during the night, the workstations are usually busy during the night as well as during the day. To avoid situations where the Ethernet network is overloaded by a large FTP or something else, each measurement is repeated twice, and the best performance is selected.

Twenty-five HP9000/700 workstations are used which are connected together by a shared-bus Ethernet network. Sixteen of the workstations are 715/50 models, six are 720 models, and three are 710 models. The 715/50 workstations are based on a Risk processor running at 50 MHz, and have an estimated performance of 62 MIPS and 13 MFLOPS, while the 720 and 710 workstations have a slightly lower performance.

For analysis purposes, we define the speed of a workstation as the number of fluid

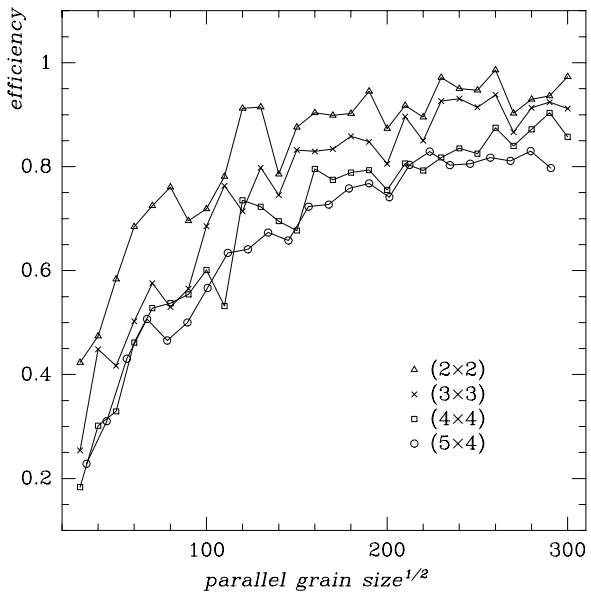

Figure 6-7: Parallel efficiency in 2D simulations using finite differences.

nodes integrated per second, where the number of fluid nodes does not include the padded areas discussed in section 6.4.2. The table below presents the speed of the workstations for 2D and 3D simulations using the lattice Boltzmann method (LB) and the finite difference method (FD). These numbers have been calculated by averaging over simulations of different size grids that range from 100^2 to 300^2 fluid nodes in 2D, and from 10^3 to 44^3 in 3D. Also, the speeds have been normalized relative to the speed of the 715/50 workstation,

	715/50	710	720
LB 2D	$1.0 \pm .04$	$.84 \pm .02$	$.86 \pm .08$
LB 3D	$.51 \pm .01$	$.40 \pm .01$	$.42 \pm .02$
FD 2D	$1.24 \pm .1$	$1.08 \pm .1$	$1.17 \pm .1$
FD 3D	$1.0 \pm .1$	$.85 \pm .1$	$.94 \pm .1$

The relative speed of 1.0 corresponds to 39132 fluid nodes integrated per second.

In the graphs of parallel speedup and efficiency, I use the 715/50 workstation to represent the single processor performance. I do not use the performance of the slowest workstation (the 710 model) for normalization purposes because it would

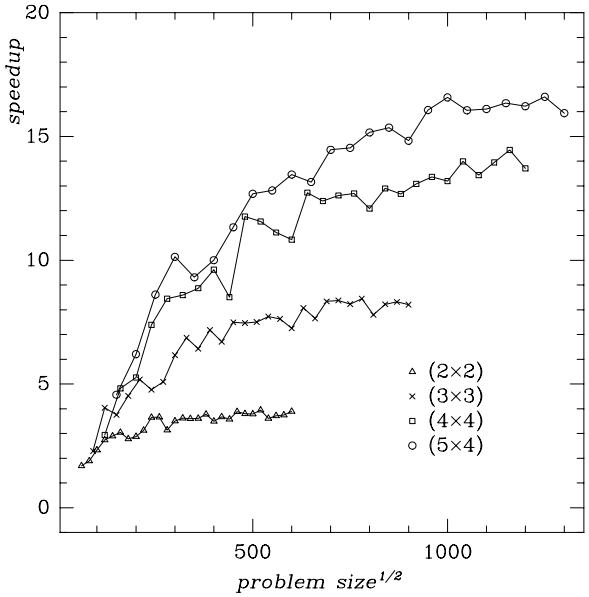

Figure 6-8: Parallel speedup in 2D simulations using finite differences.

over-estimate the performance of the system. In particular, most of the workstations are 715 models, and the strategy is to choose 715 models first before choosing the slightly slower 710 and 720 models. I have tested that the speedup achieved by sixteen workstations, which are all 715 models, does not change if one or two workstations are replaced with 710 models. Thus, it makes sense to normalize the results using the performance of the 715 model.

Figure 6-5 shows the efficiency as a function of grain size for (2×2), (3×3), (4×4), and (5×4) decompositions (triangles, crosses, squares, circles). The horizontal axis plots the square root of number of nodes N of each subregion. We see that good performance is achieved in two-dimensional simulations when the subregion per processor is larger than 100^2 fluid nodes. In the next section, a theoretical model of parallel efficiency is presented which predicts very accurately the experimental results shown in figure 6-5 and in the other figures also. Figure 6-6 shows the speedup for the lattice Boltzmann method (LB), and figures 6-7 and 6-8 show the efficiency and speedup for the finite difference method (FD).

We notice one difference between the FD and LB efficiency curves: the efficiency

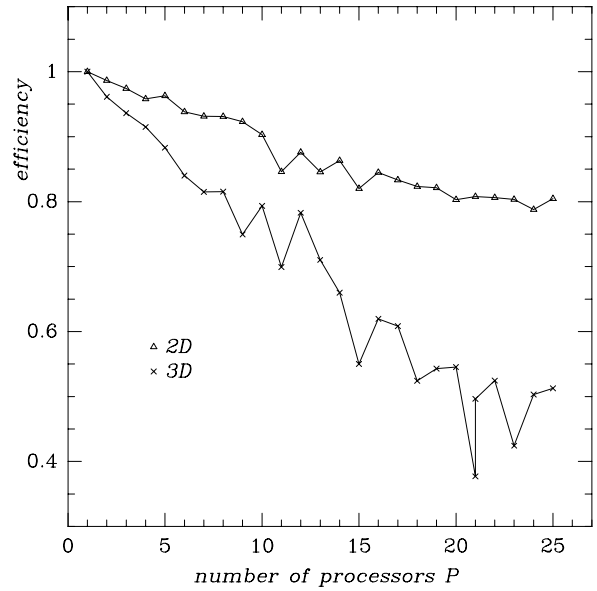

Figure 6-9: The Ethernet network performs well for 2D simulations (triangles), but poorly for 3D simulations (crosses).

decreases more rapidly for FD than LB as the subregion per processor decreases. To understand this difference, we quote a general formula for the parallel efficiency, which is derived in the next section (see equation 6.8),

$$f \;=\; \left(1 \,+\, \frac{T_{com}}{T_{calc}}\right)^{-1} \tag{6.2}$$

where T_{com} and T_{calc} are the communication and the computation time it takes to perform one integration step. We observe that T_{calc} is smaller for FD than LB (see the table of speeds earlier), and moreover that T_{com} becomes larger for FD than LB as the subregion per processor decreases. The latter is true because each message in a local-area network incurs an overhead, and FD communicates two messages per integration step as opposed to LB which communicates only one message per integration step (see end of section 6.6). Because of these differences between FD and LB, the efficiency decreases more rapidly for FD than LB as the subregion per processor decreases.

Next, we compare the efficiency of three-dimensional simulations versus two-dimensional ones. Figure 6-9 plots the efficiency of 2D and 3D simulations as a function of the number of processors P. Here, a problem is simulated which grows

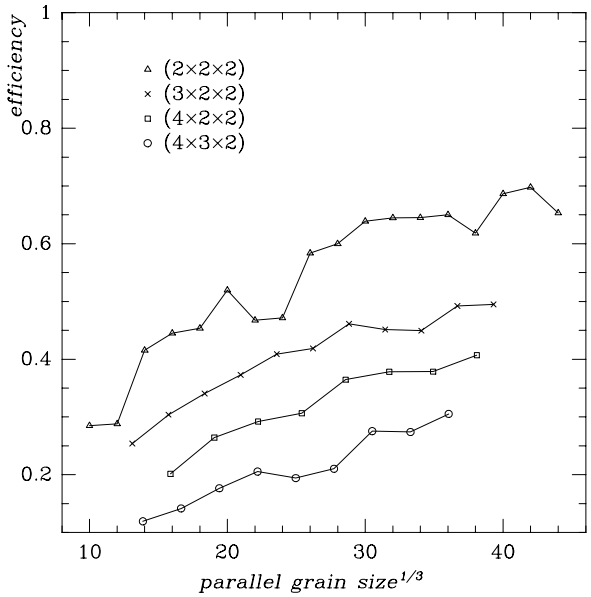

Figure 6-10: Parallel efficiency in 3D simulations using the lattice Boltzmann method.

linearly with the number of processors P, and is decomposed as $(P \times 1)$ in 2D, and as $(P \times 1 \times 1)$ in 3D. The subregion per processor is held fixed at 120^2 nodes in 2D, and 25^3 nodes in 3D, which are comparable sizes, equal to about $14,500$ fluid nodes per processor. We see that the efficiency remains high in 2D (triangles), and decreases quickly in 3D (crosses) as the number of processors increases. This is because the total traffic through the shared-bus network increases in proportion to the number of processors, and this affects T_{com} in equation 6.2 as explained in more detail in the next section. Also, we note that 3D requires much more data to be communicated per step than 2D. Thus, T_{com} increases faster for 3D than 2D, and the efficiency drops faster in the case of 3D simulations.

Another way of examining the efficiency of 3D simulations is shown in figures 6-10 and 6-11. Figure 6-10 plots the efficiency against the size of the subregion for different decompositions $(2 \times 2 \times 2)$, $(3 \times 2 \times 2)$, etc. We can see that the efficiency is rather poor. Figure 6-11 plots the speedup against the total size of the problem. We can see that the speedup does not improve when finer decompositions are employed because the network is the bottleneck of the computation.

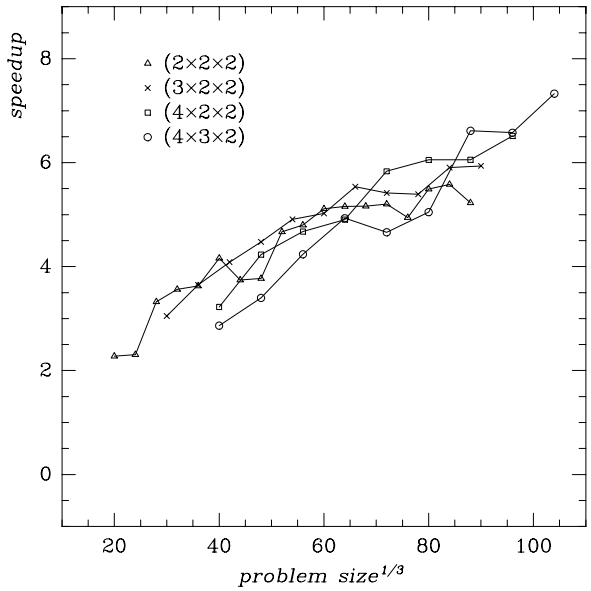

Figure 6-11: Parallel speedup in 3D simulations using the lattice Boltzmann method.

The results shown in figures 6-10 and 6-11 have been obtained using the lattice Boltzmann method. The parallel efficiency of the finite difference method (FD) in 3D simulations is even worse than the lattice Boltzmann method (LB), and is not shown here. The FD efficiency is worse than LB because the FD computes twice as fast as LB per integration step (see earlier table of speeds), which makes the ratio T_{com}/T_{calc} larger for FD than LB, and leads to lower efficiency according to equation 6.2.

Another point is that the low efficiency of 3D simulations is accompanied by frequent network errors because of excessive network traffic. In particular, the TCP/IP protocol fails to deliver messages after excessive retransmissions. Both the low efficiency, and the network errors indicate the need for a faster network, or dedicated connections between neighboring processors in order to perform 3D simulations efficiently.

6.8 Theoretical analysis of parallel efficiency

In order to understand better the experimental results of the previous section, we discuss here a theoretical model of the parallel efficiency of local-interaction problems. In particular, we derive a formula for the parallel efficiency in terms of the parallel grain size (the size of the subregion that is assigned to each processor), the speed of the processors, and the speed of the communication network. The analysis is based on two assumptions: (i) the computation is completely parallelizable, and (ii) the communication does not overlap in time with the computation. The first assumption is valid for local-interaction problems, and the second assumption is valid for the present distributed system. The extension of the analysis to situations where communication and computation overlap in time is straightforward as we shall see afterwards.

We first examine the relationship between the efficiency and the processor utilization. We define the efficiency f as the speedup S divided by the number of processors P. Further, we define the speedup S as the ratio T_1/T_p of the total time it takes to solve a problem using one processor, denoted T_1, divided by the total time it takes to solve the same problem using P processors, denoted T_p. In other words, we have the following expression,

$$f = \frac{S}{P} = \frac{T_1}{P\,T_p} \tag{6.3}$$

We define the processor utilization g as the fraction of time spent for computing, denoted T_{calc}, divided by the total time spent for solving a problem which includes both computing and waiting for communication to complete. Also, we use the simplifying assumption that the communication and the computation do not overlap in time, so that we define T_{com} as the time spent for communication without any computation occurring during this time. Thus, we have the following expression,

$$g = \frac{T_{calc}}{T_{calc} + T_{com}} = \left(1 + \frac{T_{com}}{T_{calc}}\right)^{-1} \tag{6.4}$$

To compare f and g, we note that the values of both f and g range between the

following limits,

$$\begin{aligned} 0 \leq g \leq 1 \\ 0 \leq f \leq 1 \end{aligned} \tag{6.5}$$

for the worst case and the best case respectively. We expect that high utilization g corresponds to high parallel efficiency f. However, this depends on the problem that we are trying to compute in parallel.

In the special case of a problem that is completely parallelizable, the processor utilization g is exactly equal to the parallel efficiency f. To show this, we use the following relation as the definition of a problem being completely parallelizable,

$$T_{calc} = \frac{T_1}{P} \tag{6.6}$$

Then, we also use the assumption that communication and computation do not overlap in time, so that we can obtain a second relation,

$$(T_{calc} + T_{com}) = T_p \tag{6.7}$$

By substituting equations 6.6 and 6.7 into equation 6.3, and comparing with equation 6.4, we arrive at the desired result that the parallel efficiency is exactly equal to the processor utilization,

$$f = g = \left(1 + \frac{T_{com}}{T_{calc}}\right)^{-1} \tag{6.8}$$

The above equation has been derived under the assumption that communication and computation do not overlap in time. If this assumption is violated in a practical situation, then the communication time T_{com} should be replaced with a smaller time interval, the effective communication time. This modification does not change the conclusion $f = g$, it simply gives higher values of efficiency and utilization.

To proceed further, we need to find how the ratio T_{com}/T_{calc} depends on the size of the subregion. First, we observe that T_{calc} is proportional to the size of the subregion. If N is the size of the subregion (the number of parallel nodes that constitute one

subregion), we can write,

$$T_{calc} = \frac{N}{U_{calc}} \tag{6.9}$$

where U_{calc} is a constant, the computational speed of the processors for the specific problem at hand. In a similar way, we seek to find a formula for the communication time T_{com} in terms of the size of the subregion that is assigned to each processor. As a first model, we write the following simple expression,

$$T_{com} = \frac{N_c}{U_{com}} \tag{6.10}$$

where N_c is the number of communicating nodes in each subregion, namely the outer surface of each subregion. The factor U_{com} represents the speed of the communication network.

For analysis purposes, we want to know exactly how N_c varies with the size of the subregion N. We consider the geometry of a subregion in two dimensions. We can see that the boundary of a subregion is one power smaller than the volume expressed in terms of the number of nodes. For example, if we consider square subregions of size L^2 nodes, the enclosing boundary contains $4L$ nodes, and the ratio of communicating nodes to the total number of nodes per subregion can be as large as $4/L$. In general, we have the following relations,

$$N_c = m\, N^{1/2} \tag{6.11}$$

$$N_c = m\, N^{2/3} \tag{6.12}$$

in two and three dimensions respectively, where the constant m depends on the geometry of the decomposition. For example, if the decomposition of a problem is $(P \times 1)$, then $m = 2$ because each subregion communicates with its left and right neighbors only. The following table gives m for a few decompositions which are used in the performance measurements of section 6.7,

	$P \times 1$	2×2	3×3	4×4	5×4
m	2	2	3	4	4

If we introduce the above formulas for N_c and m into equation 6.8, we obtain the following expressions for the parallel efficiency of a local-interaction problem in two and three dimensions respectively,

$$f = \left(1 + N^{-1/2} \frac{m\, U_{calc}}{U_{com}}\right)^{-1} \tag{6.13}$$

$$f = \left(1 + N^{-1/3} \frac{m\, U_{calc}}{U_{com}}\right)^{-1} \tag{6.14}$$

The above equations show that if N is sufficiently large compared to the term mU_{com}/U_{calc}, then high parallel efficiency can be achieved.

A few comments are in order. First, we must remember that in practice we can not increase arbitrarily the size of the subregion per processor in order to achieve high efficiency. This is because the computation may take too long to complete, and because the memory of each workstation is limited. In the present system, each workstation has maximum memory 32 megabytes, and a large part of this memory is taken by other programs, and other users. A practical upper limit of how much memory can be used per workstation is 15 megabytes, which corresponds to 300^2 fluid nodes in 2D simulations and 40^3 fluid nodes in 3D simulations.

In 2D simulations, the upper limit of 300^2 fluid nodes per subregion is large enough to achieve high efficiency. As we saw in figure 6-5, high efficiency is achieved when the subregion per processor is larger than 100^2 fluid nodes. By contrast, in 3D simulations the upper limit of 40^3 fluid nodes per subregion is too small to achieve high efficiency. Further, the efficiency depends on the size of the subregion as $N^{-1/3}$ in 3D versus $N^{-1/2}$ in 2D, as can be seen from equations 6.13 and 6.14. This means that the size of the subregion N must increase much faster in 3D than in 2D to achieve similar improvements in efficiency. Because of this fact, achieving high efficiency in 3D simulations is much more difficult than in 2D simulations.

Having described the basics of the model of parallel efficiency, we now discuss a small improvement of the model. We observe that in the case of a shared-bus network the communication time T_{com} must depend on the number of processors that are using

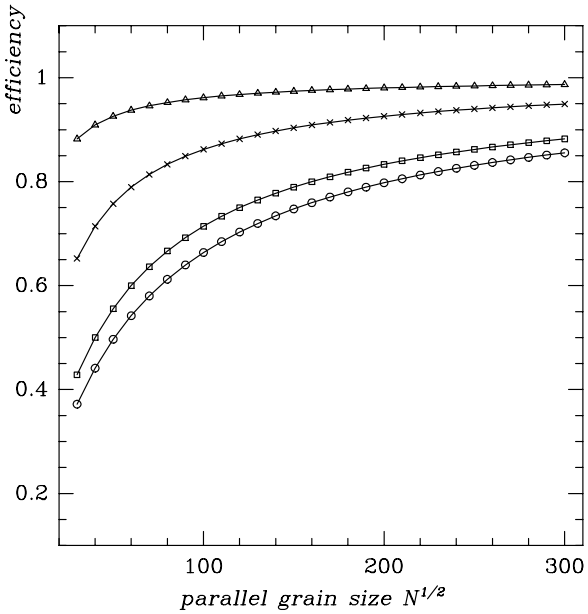

Figure 6-12: Theoretical model of parallel efficiency for two-dimensional subregions of size N.

the network. In particular, if we assume that all the processors access the shared-bus network at the same time, then the communication time T_{com} must increase linearly with the number of processors. Based on this assumption, we rewrite equation 6.10 for T_{com} as follows,

$$T_{com} \;=\; \frac{m\, N^{1/2}\,(P-1)}{V_{com}} \tag{6.15}$$

for the case of two dimensional problems. The constant V_{com} is the speed of communication when there are only two processors sharing the network. Using the new expression for T_{com}, the equation of parallel efficiency in two dimensions becomes as follows,

$$f \;=\; \left(1 \;+\; N^{-1/2}\,(P-1)\,\frac{m\,U_{calc}}{V_{com}}\right)^{-1} \tag{6.16}$$

Below, this model is tested by comparing the efficiency which is predicted by the model against the experimentally measured efficiency of section 6.7.

Figure 6-12 plots the efficiency f versus $N^{1/2}$ according to formula 6.16, using $U_{calc}/V_{com} \;= 2/3$. The four curves marked with triangle, cross, square, circle correspond to different numbers of processors $P = 4, 9, 16, 20$ and also different values

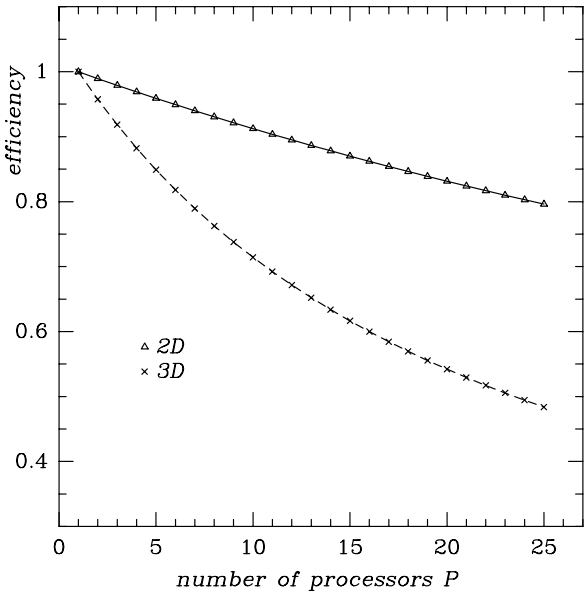

Figure 6-13: Theoretical model of parallel efficiency which assumes that the communication time increases linearly with the number of processors.

of $m = 2, 3, 4, 4$ which depends on the geometry of the decomposition as explained earlier. A comparison between the predicted efficiency shown in figure 6-12 and the experimentally measured efficiency shown in figure 6-5 reveals good agreement when the subregion per processor is larger than $N > 100^2$. However, for small subregions, $N < 100^2$, the predicted efficiency is too high compared to the experimental efficiency. The reason for this is that messages in a local-area network have a large overhead which becomes important when the messages are small, namely, when the subregion per processor is smaller than $N < 100^2$ fluid nodes. The overhead of small messages leads to a smaller communication speed V_{com}, and a corresponding decrease of efficiency f. We have not attempted to model the overhead of small messages here.

Another way of examining the validity of equation equation 6.16 is to plot the efficiency f versus the number of processors P while keeping all other parameters constant. In figure 6-13, the efficiency of 2D simulations is plotted according to equation 6.16 using $N = 125^2$. We set $U_{calc}/V_{com} = 2/3$ as we did in figure 6-12, and we set $m = 2$ because each subregion communicates with its left and right neighbors only. For comparison purposes, the efficiency of 3D simulations is also plotted, using

$N = 25^3$ and $m = 2$. The computational speed is half as large in 3D than in 2D, and the communication of each fluid node in 3D requires 5/3 as much data as in 2D. Taking these numbers into account, we can write the following expression for the parallel efficiency of 3D simulations,

$$f = \left(1 + \frac{5}{6} N^{-1/3} (P-1) \frac{m U_{calc}}{V_{com}}\right)^{-1} \tag{6.17}$$

where the factor 5/6 arises because the 2D values of U_{calc} and V_{com} are used which give $U_{calc}/V_{com} = 2/3$.

If we compare the predicted efficiency shown in figure 6-13 against the experimentally measured efficiency shown in figure 6-9, we can see that there is good agreement. Also, the overhead of small messages, mentioned earlier, does not affect the predicted efficiency in this case because the subregion per processor is large, $N = 125^2$ in 2D, and 25^3 in 3D. Overall, there is reasonable agreement between the theoretical model and the experimental measurements of parallel efficiency. The model can be improved further, if desired, by employing more sophisticated expressions for the communication time T_{com} in equation 6.15 which describes the behavior of the shared-bus Ethernet network.

6.9 Conclusion

An effective approach of simulating fluid dynamics on a cluster of non-dedicated workstations has been presented. The approach is particularly good for simulating subsonic flows which involve both hydrodynamics and acoustic waves. A parallel simulation system has been developed and applied to solve a real problem, the direct simulation of flue pipes of wind musical instruments.

The system achieves concurrency by decomposing the flow problem into subregions, and by assigning the subregions to parallel processes. The use of explicit numerical methods leads to minimum communication requirements. The parallel processes automatically migrate from busy hosts to free hosts in order to exploit

the unused cycles of non-dedicated workstations, and to avoid disturbing the regular users. Typical simulations achieve 80% parallel efficiency (speedup/processors) using 20 HP-Apollo workstations.

Detailed measurements of the parallel efficiency of 2D and 3D simulations have been presented, and a theoretical model of efficiency has been developed which fits closely the measurements. The measurements show that a shared-bus Ethernet network with 10Mbps peak bandwidth (megabits per second) is sufficient for two-dimensional simulations of subsonic flow, but is limited for three-dimensional simulations. It is expected that the use of new technologies in the near future such as Ethernet switches, FDDI and ATM networks will make practical three-dimensional simulations of subsonic flow on a cluster of workstations.

6.10 Appendix

The appendix describes certain aspects of the distributed system that are not vital for a general reading, but are useful to someone who is interested in implementing a distributed system similar to the present one.

6.10.1 Synchronization issues

The synchronization between distributed processes (see section 6.4.2) can be violated in situations such as the following. Let us suppose that process A stops execution after communicating its data for integration step N. The nearest neighbor B can integrate up to step $N+1$ and then stop. Process B can not integrate any further without receiving data for integration step $N+1$ from process A. However, the next to nearest neighbor can integrate up to step $N+2$, and so on. If we consider a two-dimensional decomposition $(J \times K)$ of a problem, the largest difference in integration step between two processes is ΔN,

$$\Delta N = max(J, K) - 1 \tag{6.18}$$

assuming that neighbors depend on each other along the diagonal direction (this corresponds to a full stencil of local interactions as shown in figure 6-4). If neighbors depend on each other along the horizontal and vertical directions only (this is the star stencil of figure 6-4), then the largest difference in integration step between two processes becomes,

$$\Delta N = (J-1) + (K-1) \tag{6.19}$$

These worst cases of un-synchronization are important during the migration of processes because a precise global synchronization is required then, as explained in section 6.5.

The synchronization algorithm that is used during process migration is as follows. First, we send a synchronization request to all the processes by means of a UNIX interrupt. In response to the request, every process writes the current integration time step into a shared file (using file locking semaphores, and append mode). Then, every process examines the shared file to find the largest integration time step $T_{\max}$ among all the processes. Further, every process chooses ($T_{\max}$+1) to be the upcoming synchronization time step, and continues running until it reaches this time step. It is important that all the processes can reach the synchronization time step, and that no process continues past the synchronization time step.

The above algorithm finds the smallest synchronization time step that is possible at any given time, so that a pending migration can take place as soon as possible.

6.10.2 Alternative communication mechanisms

A minor efficiency issue with regard to TCP/IP communication (see section 6.4.2) is the order in which the neighboring processes communicate with each other. One way is for each parallel process to communicate with its neighbors on a first-come-first-served basis. An alternative way is to impose a strict ordering on the way the processes communicate with each other. For example, we consider a one-dimensional decomposition ($J \times 1$) of a problem with non-periodic outer-boundaries where each

process receives data from its left neighbor before it can send data to its right neighbor. Then, the leftmost process No. 1 will access the network first, and the nearest-neighbor process No. 2 will access the network second, and so on. The intent of such ordering is to pipeline the messages through the shared-bus network in a strict fashion in an attempt to improve performance. However, it does not work very well if one process is delayed because all the other processes are delayed also. Small delays are inevitable in time-sharing UNIX systems, and strict ordering amplifies them to global delays. By contrast, asynchronous first-come-first-served communication allows the computation to proceed in those processes that are not delayed, and better performance is achieved overall. In the parallel system, first-come-first-served communication is implemented using the "select" system call of sockets (see UNIX manual).

Regarding the choice of communication protocol, the TCP/IP protocol is used because it is very simple as explained in section 6.4.2. Apart from the TCP/IP protocol, another protocol that is popular in distributed systems is the UDP/IP protocol, also known as datagrams. The UDP/IP protocol is similar to TCP/IP with one major difference: There is no guaranteed delivery of messages. Thus, the distributed program must check that messages are delivered, and resend messages if necessary, which is a considerable effort. However, the benefit is that the distributed program has more control of the communication. For example, a distributed program could take advantage of knowing the special properties of its own communication to achieve better results than the TCP/IP standard. Also, another advantage is robustness in the case of network errors that occur under very high network traffic. For example, when TCP/IP fails, it is hard to know which messages need to be resent. In UDP/IP the distributed program controls precisely which data is sent and when, so that the failure problem is handled directly. Despite these advantages of UDP/IP over TCP/IP, I have chosen to work with TCP/IP because of its simplicity.

6.10.3 Performance bugs to avoid

In section 6.7, we examined the performance of the HP-Apollo workstations. It should be noted that the performance of the HP9000/700 Apollo workstations can degrade dramatically at certain grid sizes by a factor of two or more, but there is an easy way to fix the problem. The loss of performance occurs when the length of the arrays in the program is a near multiple of 4096 bytes which is also the virtual-memory page size. This suggests that the loss of performance is related to the prefetching algorithm of the CPU cache of the HP9000/700 computers. To avoid the loss of performance, the arrays can be lengthened with 200-300 bytes when their length is a near multiple of 4096. This modification eliminates the loss of performance.

Another problem that can lead to loss of performance is the handling of floating-point exceptions. When an underflow exception occurs, the HP9000/700 workstations trap into the system kernel by default, and this causes considerable slow-down. The slow-down is amplified in a distributed computation because if one processor slows down, all the processors slow down. A particular situation in fluid dynamics occurs when the passage of an acoustic wave causes underflow exceptions to different processors at different times. Then, during the passage of the acoustic disturbance, all the processors are delayed. Such problems can be observed at the beginning of the simulation when the fluid begins to move from an initial non-moving state (namely, the density variations of the fluid are equal exactly to zero at startup). Fortunately, there is a simple solution which is to avoid initializing the fluid density variations equal to exact zero; for example, an initial density gradient with relative size $10^{-}10$ is practically the same as zero in the present situation. Such a non-zero initialization avoids the floating-point underflow. Another solution which is available in the HP9000/715 workstation models but not in the 720 models, is to set "fast underflow mode" using the system call "fpsetfastmode" of HPUX. Fastmode causes the hardware to simply substitute a zero for the result of an operation that underflows, without a system fault and without any delay.

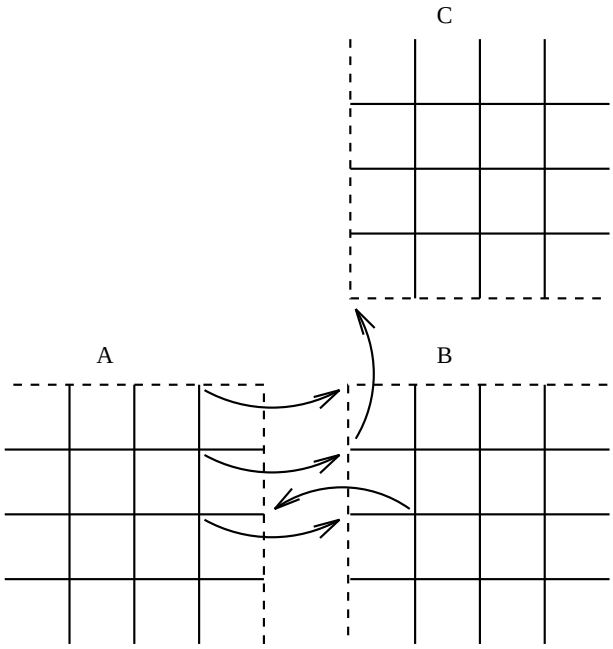

Figure 6-14: Communication of data across boundaries (dashed lines) for the finite difference method.

6.10.4 Communication of fluid flow boundaries

In section 6.6, the parallelization of explicit numerical methods for fluid dynamics was discussed. Here, the precise manner in which the fluid flow boundaries are communicated between the parallel processes is described. The finite difference method communicates the fluid variables (ρ, V_x, V_y) in 2D, and (ρ, V_x, V_y, V_z) in 3D. The lattice Boltzmann method communicates the moving populations F_i that must be shifted across a boundary. There are 3 moving populations F_i in each direction in 2D, and 5 moving populations F_i in each direction in 3D.

Figures 6-14 and 6-15 show how the boundary values are communicated along the x and y directions. In the case of the finite difference method (figure 6-14), the values on the inner nodes next to the padded area of region A are copied onto the padded area of region B. In the case of the lattice Boltzmann method (figure 6-15) the values on the padded area of region A are copied onto the inner nodes of region B. The differences

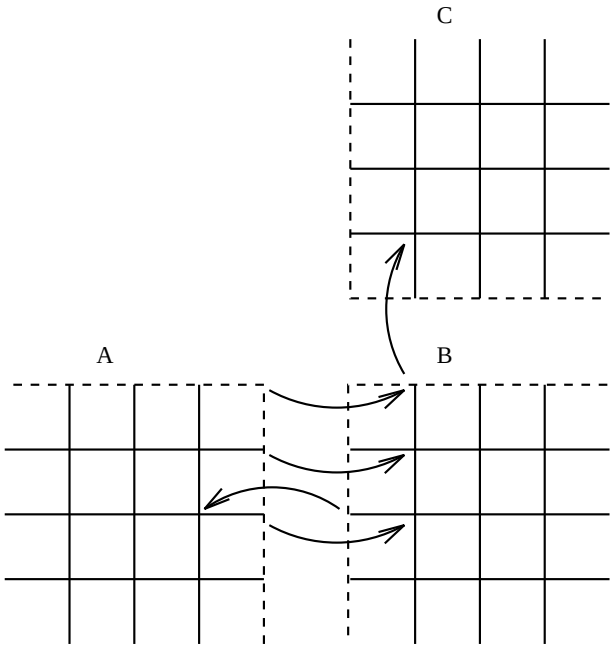

Figure 6-15: Communication of data across boundaries (dashed lines) for the lattice Boltzmann method.

in data movement are due to the fact that the finite difference method communicates the fluid variables ρ, V_x, V_y, while the lattice Boltzmann method communicates the moving populations F_i. The moving populations F_i are shifted to the padded areas prior to the communication operations.

The corner nodes of each rectangular region need special attention because they connect regions diagonally (for example, regions A and C in figure 6-14). A simple way of handling diagonal connections is to communicate along the x-direction first, then along the y-direction, then along the z-direction. Thus, the diagonal corner values are updated correctly at the expense of constraining the order of communication. The lattice Boltzmann method obeys this constraint. The finite difference method however does not obey this constraint, and it ignores the corner points. This is a special case because in the present simulations the differencing stencils are cross-shaped without diagonals. This is exploited so that the communication operations of

the finite difference method take place in any order between the x,y,z directions, and there are no diagonal dependencies.

Chapter 7

Music by flue pipes

First, I review flow-generated sound phenomena in general, and then I focus on simulations of flue pipes. The results presented here are a continuation of the computer simulations and physical measurements already described in chapter 1.

7.1 Background

7.1.1 Related computational work

Related work on simulating flow-generated sound phenomena has been limited, and all of the previous studies have employed incompressible flow equations as far as I know. For example, Ohring [35] has simulated jets of air impinging on a sharp triangular wedge using an incompressible flow calculation. Peters [37] has employed vortex methods to simulate the initial stages of blowing air through a flue channel and also the flow of gas through industrial pipe systems. Harding [24] has used an incompressible flow calculation as a source term to a wave equation in order to study the sound generated by an obstruction inside a channel. As explained earlier in chapter 1, Harding's approach applies only when the acoustic waves do not interact with the hydrodynamic flow. In the case of flue pipes, acoustics and hydrodynamics must be simulated together using the compressible Navier Stokes equations.

7.1.2 Catalogue of flow-generated sound phenomena

There is a very wide variety of sound phenomena which are triggered and sustained by the flow of air (or any fluid medium) and the interaction between the flow and solid obstacles. The following are some well-known examples.

- Flue pipes exploit the oscillations of narrow jets of air that impinge a sharp obstacle called the labium. The operation of flue pipes depends on the coupling between acoustic and hydrodynamic oscillations. Flue-based musical instruments include the baroque recorders, the flutes, the organ pipes found in cathedrals, the pipes used by Latin America cultures, the pan-pipes of ancient Greece, and the bamboo flutes found in many Pharaoh's tombs inside Egyptian pyramids.

- A Helmholtz resonator (a glass bottle) can be used in the place of a long pipe. Blowing a narrow jet of air over the opening of a bottle generates pure tones of a definite frequency.

- The sound generated by swinging around a plastic tube with a diameter of $1-5$ cm (children often do this) is probably similar to blowing air over a pipe or a bottle. An observer that stays with the moving tube sees the air rushing over the opening of the tube. The layer of air (boundary layer) next to the opening of the tube is very unstable, and can easily start oscillating near the resonant acoustic frequencies of the pipe.

- Whistles are close relatives to flue pipes. For instance, in human whistling, the teeth and the tongue are used to form a narrow jet of air. The jet of air is blown against an obstruction of appropriate shape (the lips). The mouth probably acts as a resonator in this case.

- Another type of whistling (lower frequency than lip-whistling) is possible by putting one's hands together to form a cavity with a narrow opening between

the two thumbs, and by blowing a narrow jet of air (using one's lips and teeth) tangentially onto the opening between the two thumbs. The thumb-nails should be positioned below one's nose in order to blow air tangentially onto the opening between the thumbs. If this has not been done before, it takes some experimentation at first to get it right.

- Besides flue pipes and whistles, another flow-generated sound phenomenon is the Aeolian tone. An Aeolian tone is generated when a stream of air flows around a narrow obstacle, such as a wire or a cylinder. The stream of air may be wide, or it may be very narrow so that it can be viewed as a jet of air. Morse&Ingard [33, p.751] provide experimental formulas for the frequencies of Aeolian tones as a function of air speed and wire diameter. A related musical instrument is the Aeolian harp which consists of a set of strings that vibrate when the air is blown against them.

- In sound phenomena such as Aeolian tones, the acoustic and the hydrodynamic oscillations of the air typically trigger vibrations of the wire so that there is a coupling between acoustic, hydrodynamic, and solid-obstacle oscillations. This coupling amplifies the resonant frequencies of the wire, and sometimes it even leads to disaster when there is not sufficient damping of the solid-object vibrations. The collapse of the Tacoma bridge and the collapse of industrial chimneys (Tritton [54, p.444]) are famous examples.

- Reed musical instruments such as the clarinet and the harmonica also exploit the vibrations of solid obstacles. It should be noted however that a vibrating reed is somewhat different from a vibrating wire because the reed vibrations open and close periodically a narrow opening through which the air passes.

The above catalogue describes some representative examples of flow-generated sound phenomena. Many other possibilities and variations of the above are certainly possible. Below, the operation of flue pipes is considered further.

7.2 The operation of flue pipes

The operation of flue pipes has been studied for hundreds of years. Considerable progress has been made, but important basic questions remain unanswered. For example, a most basic question is whether a given geometry (flue channel, labium, and pipe) will produce audible tones. Anyone who has experimented with building new kinds of flue pipes knows very well that small changes in the geometry can make a flue pipe sing, make no sound at all, or make a very mediocre sound (noisy, hissing, or including intermittent vibrations and beats). Presently, the existing theories of flue pipes can not answer questions such as whether a given flue pipe will sing or not.

The existing theories of flue pipes try to reduce the complexities of the fluid dynamics inside a flue pipe to a system of lumped components such as oscillators (inductors, capacitors), dampers, and amplifiers. Such a reduction introduces a number of parameters which are adjusted to fit the observed results of a particular flue pipe (Verge94 [57, 56], Hirschberg [26]). Considerable success has been reported with some reduced models of flue pipes, but the subject still has a long way to go. For example, the assumptions of the reduced models are not agreed upon by everyone, and they are not completely understood. Furthermore, finding reduced models of flue pipes is somewhat of an art. It is not clear what approximations can be made when a new flue pipe of different geometry is considered.

The details of existing theories of flue pipes will not be discussed here. However, there are a few basic principles that are worth reviewing. First, it is assumed that there is some kind of feedback between the acoustic waves in the pipe and the jet. This feedback is responsible for amplifying the acoustic waves under appropriate conditions. Second, it is recognized that there are, at least, two major types of feedback: hydrodynamic and acoustic. The hydrodynamic feedback refers to the interaction between the jet of air and the labium, and includes the shedding of vortices by the jet, and the local pressure gradients which have an immediate effect on the jet. The acoustic feedback refers to the pressure disturbances (traveling waves) which

emanate from the jet-labium region, travel down the pipe, reflect, and return back to the jet-labium region after a considerable delay. The major distinction between acoustic and hydrodynamic feedback is the time delay of traveling waves versus the almost-zero delay of hydrodynamic effects.

The distinction between the hydrodynamic and the acoustic feedback is closely related to the distinction between an "edge tone" and a "pipe tone". The former refers to the oscillations of a jet impinging a sharp obstacle without any resonant cavity in the vicinity. The latter, the pipe tone, refers to the normal operation of a flue pipe, where sound is generated by a jet of air impinging a sharp edge near a resonant pipe. It is clear that in the edge tone there is no reflection of acoustic waves (no delayed feedback) which means that the edge tone is a purely hydrodynamic phenomenon by definition. Furthermore, the frequencies of an edge tone are approximately proportional to the blowing speed, and inversely proportional to the distance between the jet's orifice and the obstacle. An experimental formula is as follows (Hirschberg [26, p.210]),

$$\frac{f\,W}{V} \;=\; 0.4\,(n+\gamma) \qquad n = 1, 2, \ldots \tag{7.1}$$

where f is the frequency in Hz, V is the mean speed of the jet in cm/s, and γ is a small correction $0 \leq \gamma \leq 0.5$. By contrast, the frequencies of a pipe tone do not vary much with the blowing speed (except for jumping to higher modes), and are determined mostly by the acoustic feedback and the dimensions of the resonant pipe. As the blowing speed increases, the pipe-tone frequencies stay approximately fixed until at some point the frequencies "jump" to higher values which are near higher resonant modes of the pipe. This is, of course, a simplified picture. In practice, low-frequency beats, hissing sound, and failure to sing may also occur as the blowing conditions are varied.

In comparing edge tones and pipe tones, it should be noted that an edge tone often does not generate enough acoustic energy to be audible. Generally, an edge tone is weaker than a pipe tone because there is no resonant cavity to amplify the sound. A

related issue is that when a flue pipe stops singing, the jet of air often continues to oscillate. Perhaps, this is a type of edge tone where the acoustic coupling between the jet and the resonant cavity fails for some reason, and the hydrodynamic effects play a dominant role in the jet's oscillations, but do not generate enough acoustic energy to be audible (figure 7-9 shows a simulation where this phenomenon may be occurring).

The details of the hydrodynamic and the acoustic feedback are still a subject of research. A currently popular model of the acoustic feedback is to assume that the jet behaves as if it were infinitely long, and that the acoustic waves inside the pipe perturb the jet as it emerges from the flue channel. As the jet undulates, it amplifies the perturbations, and returns acoustic energy into the pipe. This model is based on the work of Rayleigh (J.W. Strutt) on infinitely long jets. Although there is some truth to this model, the actual jet inside a flue pipe is nothing but infinite. The jet is short and rather unpredictable. Sometimes, the jet extends undulating all the way from the orifice to the labium, and other times, the jet breaks well-before reaching the tip of the labium. Perhaps, different reduced models of the jet are needed to characterize different behaviors.

Some factors which control the operation of a flue pipe, what frequencies are generated, and how well the flue pipe sings are listed below.

- The blowing speed of the jet.
- The initial blow of air into the pipe that triggers the oscillations.
- The orifice-to-labium distance.
- The alignment of the labium with the flue channel, and also the alignment of the labium with the resonant pipe.
- The length of the resonant pipe, as well as the width (and depth) of the pipe.

- The conditions outside the pipe and especially above the labium. For example, an infinite region above the labium, stagnant air, and constant ambient pressure seem to help the operation of the flue pipe. By contrast, a limited region above the labium, accumulation of vorticity, and buildup of pressure gradients complicate the operation of the flue pipe.

The last one of the above conditions has already been mentioned in section 1.4 as a possible cause for the differences between the computer simulations and the physical measurements of the 20 cm closed-end soprano recorder. Below, the simulation of flue pipes is discussed further.

7.3 Inlet and outlet boundary conditions

In this section, suitable boundary conditions for modeling the inlet and the outlet in simulations of flue pipes are described. The same approach applies both to the lattice Boltzmann method and the compressible finite difference method of section 3.3.

The boundary conditions at the inlet and the outlet must ensure that a prescribed flow of air enters and exits the simulated region. Furthermore, the boundary conditions at the inlet and the outlet must avoid the reflection of acoustic waves, if possible. This is an important issue in modeling flue pipes because the region above the labium should approximate as much as possible an infinite region, not a resonant cavity.

A simple technique for non-reflecting (absorbing) boundary conditions can be devised as follows. We observe that in compressible flow, the propagation of acoustic waves occurs by interchanging the acoustic energy between two forms, kinetic (velocity) and potential (density). If either the velocity or the density is "clamped" down at a point, acoustic reflection occurs at that point. If both the velocity and the density are free to vary (as in free space), the acoustic wave propagates freely without reflections. If both the velocity and the density are "clamped" down, the acoustic wave is absorbed, and there are no reflections.

The above rules for the reflection of acoustic waves can be verified by considering a few simple cases. An example where the velocity is clamped and the density is free, is a non-slip wall. As a traveling wave reaches the wall, the acoustic velocity must vanish, which causes the density to build up at the wall, and subsequently creates a traveling wave in the opposite direction (the reflection). If the traveling wave is a pulse of positive density, so is the reflected wave; in other words, the phase of the acoustic wave is preserved after a wall reflection.

By contrast, when the velocity is free and the density is clamped, the phase of the traveling wave is reversed. An example is the reflection at the end of an open pipe; namely, a pipe which opens into infinite space. In this case, the density is held approximately constant (ambient atmospheric pressure), and the velocity varies. As the traveling wave reaches the opening, the density pulse (let us assume a positive pulse) must vanish, which causes the acoustic velocity to increase further (the potential energy becomes kinetic) until eventually a negative pulse of density is created which travels backwards (the reflection).

The above rules describe what happens in the physical world. Similar rules to the above can be applied in a numerical simulation of compressible flow.

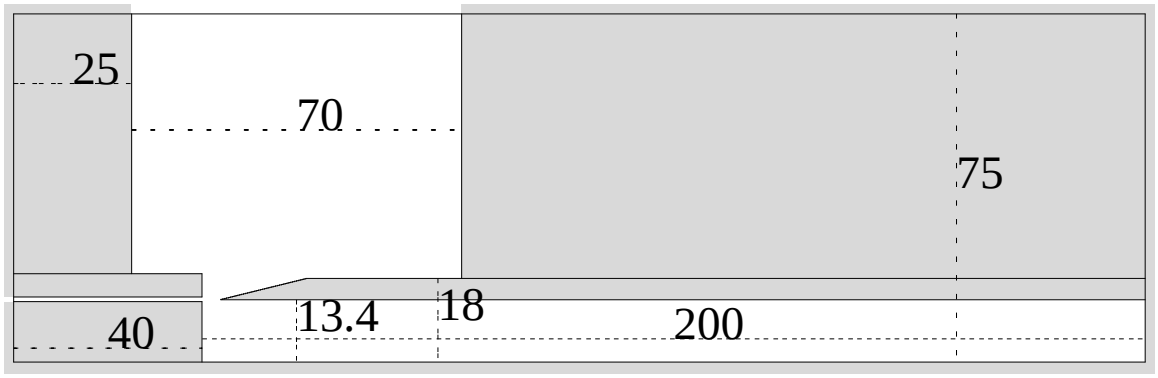

Figure 7-1: Soprano recorder flue, 20 cm closed-end pipe. The numbers shown correspond to millimeters. Inlet is at the left, outlet is at the top of the picture.

For example, in the simulation of a closed-end flue pipe, both the pressure and

the velocity are prescribed at the inlet and the outlet so as to avoid the reflection of acoustic waves. In particular, the pressure is set equal to zero at the outlet, and equal to an estimated pressure drop at the inlet. Figure 7-1 shows a typical geometry of such a flue pipe simulation. The inlet is located at the narrow opening of the flue channel at the left side, and the outlet is located at the top of the picture. It must be noted that the pressure drop is not known a priori because it depends on the imposed flow of air, and on the dynamical behavior of the system. Thus, the imposed pressure drop must be an approximation. Such an approach is successful in preventing the reflection of acoustic waves, [1] but raises the question whether it is consistent to specify both the velocity and the pressure at the inlet and the outlet.

To answer the above question, let us consider the case of Hagen-Poiseuille flow through a long pipe. When a pressure drop is imposed between the inlet and the outlet, a flow develops through the pipe. When a flow is imposed through the pipe, a pressure drop develops. When both a flow and a pressure drop are imposed, a flow develops which is higher than the imposed flow if the imposed pressure drop is an overestimate of the pressure drop corresponding to the imposed flow; and conversely if the imposed pressure drop is an underestimate. This behavior is easily verified in simulations of Hagen-Poiseuille flow and also in simulations of flue pipes using the lattice Boltzmann and the compressible finite difference method (figures 7-1A and 7-1B).

Table 7.1 shows the imposed velocity and the actual flow through the flue channel in simulations of the 20 cm closed-end recorder. The profile of the imposed velocity is

[1] Another issue which relates to hydrodynamics as opposed to acoustics is the "reflection" of vortices reaching the outlet. In particular, vortices are generated at the labium of the flue pipe, and eventually reach the outlet if the simulation continues long enough. When this happens, the vortices do not simply cross the outlet and leave the simulated region. Instead, the vortices reach the outlet, try to leave the simulated region, and then bounce back into the simulated region. The accumulation of vorticity in the simulated region creates problems because it changes the nature of the problem being simulated. This issue is avoided in the present simulations by making the simulated region large enough that the vortices generated at the labium do not reach the outlet during the simulation. Better boundary conditions or some way to dissipate the vorticity before reaching the outlet, must be devised in order to continue the simulations indefinitely.

Lattice Boltzmann				
imposed V	800	1080	1500	1959
actual V	818	1104	1535	1995
Finite Differences				
imposed V	800	1060	1558	1985
actual V	838	1113	1634	2082

Table 7.1: Imposed velocity and actual flow through the flue channel in flue pipe simulations. The velocity V is in cm/s.

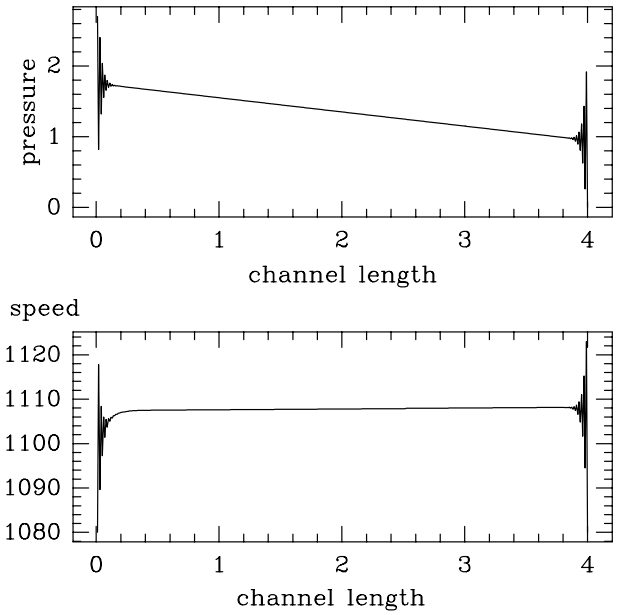

Figure 7-1A: Pressure drop and flow speed through a channel. Overimposed pressure boundary conditions between inlet and outlet. Compressible finite difference method.

parabolic both at the inlet and the outlet (the total flux at the outlet is set equal to the total flux at the inlet). The actual flow through the flue channel is measured by sampling midway along the width of the flue channel and time-averaging. The velocity profile inside the channel is parabolic so the horizontal velocity at the midpoint is scaled by 2/3 to calculate the mean speed shown in table 7.1. We can see that the actual flow is always larger than the imposed flow. This is because the imposed pressure drop is an overestimate of the pressure drop corresponding to the imposed flow through a channel 0.1 cm wide and 4 cm long. Specifically, the imposed pressure

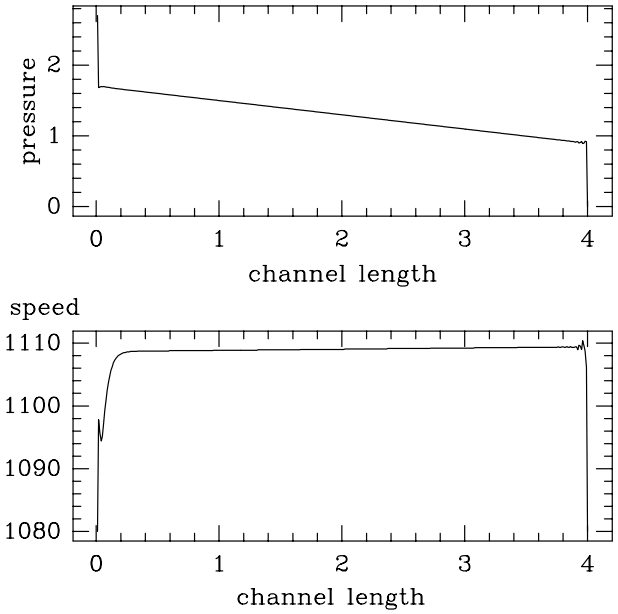

Figure 7-1B: Pressure drop and flow speed through a channel. Overimposed pressure boundary conditions between inlet and outlet. Lattice Boltzmann method using second-order differences at the boundary.

drop is equal to the Hagen-Poiseuille pressure drop of a channel whose length is 3.5 times the length of the flue channel; namely,

$$\frac{\Delta P}{\rho_0} \;=\; \Delta L\,\frac{12\nu}{d^2}\,V \;=\; (3.5\times 4.0)\times\frac{12\times 0.15}{0.1^2}\times V \;=\; 2500\times V \qquad (7.2)$$

where V is the mean velocity, ΔP is the pressure drop in $\mathrm{gm/(cm\,s^2)}$, ρ_0 is the mean density of air, and d and ΔL represent the channel's width and length.

The above pressure drop is much larger than necessary. The actual pressure drop between the inlet and the outlet of the simulations of flue pipes is dominated by the pressure drop along the narrow flue channel. Thus, it would suffice to impose a pressure drop equal to the Hagen-Poiseuille pressure drop of the flue channel. Due to an oversight (see footnote of page 225B), the pressure drop was imposed 3.5 times larger than necessary. However, an overestimated pressure drop does *not* cause any serious problems in the simulations; it only produces a slightly larger flow than the imposed flow as shown in table 7.1 and figures 7-1A and 7-1B. In general, only an order-of-magnitude estimate of the pressure drop is needed.

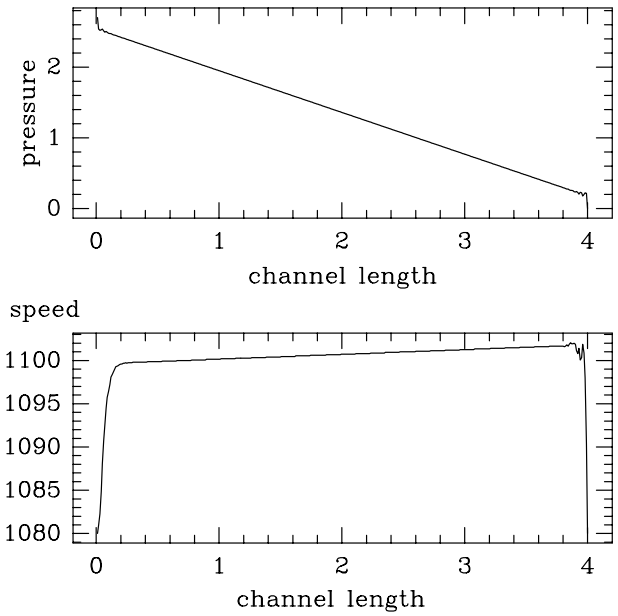

Figure 7-1C: Pressure drop and flow speed through a channel. Lattice Boltzmann method using first-order differences at the boundary.

Figures 7-1A to 7-1C show the pressure drop and flow speed during steady state in simulations of a channel which is 0.1 cm wide and 4 cm long. Both the density and velocity are imposed at the inlet and outlet, and the walls are non-slip. The setup is similar to the simulations of flue pipes except that only the channel is considered here for simplicity. The grid is 401×11. The flow speed in the figures is expressed in cm/s and the pressure drop in $c_s^2\,(\rho - \rho_0)/\rho_0$ where ρ_0 is the mean density of air, and c_s is the speed of sound. Both the pressure and the flow speed are sampled at the midpoint and along the length of the flue channel (the speed at the midpoint is scaled by 2/3 to calculate the mean speed because the velocity profile is parabolic).

Figure 7-1A corresponds to the compressible finite difference method using first-order differences at the boundary (section 3.3.4); and figures 7-1B and 7-1C correspond to the lattice Boltzmann method using second-order differences and first-order differences at the boundary respectively. In figures 7-1A and 7-1B we can see that imposing a larger-than-necessary pressure drop between the inlet and outlet, simply shifts the pressure field upwards (the curve becomes centered between the imposed

pressure values). Further, we can see that a larger-than-necessary pressure drop causes a slight increase of the flow through the channel. The change from the imposed boundary condition to the flow behavior inside the channel includes a ringing effect which is more noticeable in the case of the compressible finite difference method.

In figure 7-1C, we see that the lattice Boltzmann method using first-order differences at the boundary predicts a very different pressure drop than the results of figures 7-1A and 7-1B. In fact, the pressure-gradient slope of figure 7-1C is 3 times larger than the pressure-gradient slopes of figures 7-1A and 7-1B. It turns out that the lattice Boltzmann method using first-order differences at the boundary is very inaccurate with regard to the pressure drop, and overestimates the pressure drop by a factor of 3 at the present resolution of 10 fluid nodes per width of the channel. By contrast, the pressure drop of figures 7-1A and 7-1B agrees within 2 decimal digits with the correct value of Hagen-Poiseuille flow. [2]

An important fact to mention is that I discovered the inaccurate prediction of the pressure drop by the lattice Boltzmann method using first-order differences at the boundary, after most of the simulations presented in my thesis had already been performed. [3] Fortunately, the lattice Boltzmann simulations using first-order and second-order differences at the boundary do not differ greatly with regard to the operation of the flue pipe; they only differ with regard to the pressure drop inside the flue channel. This fact was checked for a number of different simulations. Because of this fact and because of lack of time, the lattice Boltzmann simulations which were performed using first-order differences, have not been repeated using second-order differences at the boundary. Of course, second-order differences at the boundary are recommended and should be used in the future.

[2] An explanation of the large error in pressure drop by the lattice Boltzmann method when using first-order differences at the boundary must involve the Chapman-Enskog expansion of the extended collision operator, and is left for future work.

[3] The overestimated pressure drop has been used as a boundary condition for all the simulations (lattice Boltzmann method using first-order and second-order differences at the boundary, and compressible finite difference method).

7.3.1 The end-correction of an open-end pipe

The rules mentioned in the previous section for the reflection of acoustic waves can be used to model an open-end pipe. Normally, an open-end pipe requires the simulation of a very large region connected to the outside of the open-end pipe. To save on computational effort, a shortcut can be made by imposing a fixed ambient pressure at the end of the pipe, and calculating the velocity via extrapolation. This approach reflects acoustic waves in a similar way that a physical open-end pipe does. Section 7.5 presents simulations of an open-end soprano recorder using this approach.

An issue with the above approach is the end-correction of an open-end pipe (Rayleigh [42, p.287], Olson [36, p.84]). In the physical world, the point where the pressure equals the ambient pressure is not exactly the end of the pipe, but varies depending on the diameter of the pipe and possibly on other factors as well. A related issue is that a specific amount of acoustic energy is radiated outwards during reflection from an open-end. This loss of acoustic energy may differ between the physical world and the simple model of clamping the pressure and extrapolating the velocity. These are some of the difficulties which make the modeling of an open-end pipe more difficult than the modeling of a closed-end pipe, and should be addressed in the future.

7.3.2 Smooth rise at startup

During the initial blowing of the air into the flue channel, the imposed density and velocity at the inlet rise smoothly to final values within a specified time interval. The following formula is used both for the velocity and the density,

$$V(t) = V_{\text{final}} \left[1 \; - \; 10^{-\,(t/T)^2}\right] \tag{7.3}$$

where T is the rise time it takes to reach 90% of the final value. A rise time of 3 ms is used in all the simulations presented here, which is relatively fast but not unusual (Verge94 [57, 56], Hirschberg [26]).

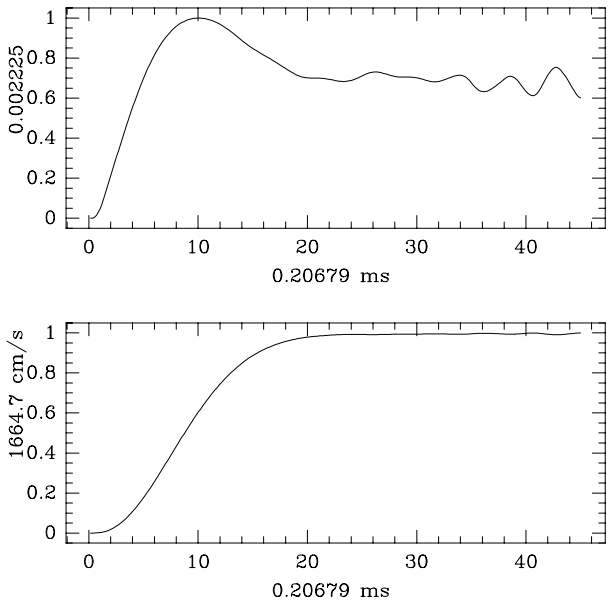

Figure 7-2: The rise of density and velocity inside the flue channel at startup.

Figure 7-2 shows the rise of the density and the velocity (x-component of velocity) inside the flue channel during the initial blowing of air. These signals are obtained from a lattice Boltzmann simulation of a closed-end recorder with a mean blowing speed 1104 cm/s (same as figure 1-16). The signals are sampled at the midpoint inside the flue channel (maximum flow velocity) and at distance 0.961 cm from the inlet. The density (shown as ρ'/ρ_0) rises at a faster rate than the velocity because the flow creates additional pressure during startup. The additional pressure is a reaction of the stagnant air inside the channel to the incoming flow. After a time interval of 20×0.206 ms, both the pressure and the velocity reach final values approximately. After 40×0.206 ms, the onset of periodic acoustic oscillations can be seen as well. The acoustic oscillations are generated at the flue-labium region, and travel backwards into the flue channel.

7.4 Closed-end soprano recorder

This section presents further results on the simulations of the closed-end soprano recorder described in section 1.4.

It is interesting that if we sample the acoustic signal (density variations) below the labium, the fundamental mode is strongly diminished compared to sampling the radiated signal outside the pipe. Most-likely, this is because the open end (flue-labium region) acts as a node for density oscillations, and an anti-node (loop) for velocity oscillations. To be precise, the flue-labium region is actually driving the oscillations, and thus it is somewhat different from an exact node of a passive pipe. Nevertheless, the flue-labium region behaves very much as an open end, and as a density node for the fundamental frequency. The effect can be observed in the computer simulations both for a closed-end recorder (open-closed pipe) and for an open-end recorder (open-open pipe) described in section 7.5.

Figures 7-6 and 7-7 show the acoustic signal (density variations) from the lattice Boltzmann simulation of a 20 cm closed-end recorder at blowing speed 1535 cm/s (same plotting conventions as in section 1.4, figure 7-6 is identical to figure 1-17). Two different sampling locations are examined: the top graphs show the signal outside the pipe and about 5 cm above the labium, the bottom graphs show the signal inside the pipe and 1.34 cm below the labium (right on the bottom wall and 0.316 cm forwards in the horizontal direction from the flue orifice). We can see that the fundamental mode of 400 Hz is diminished in the bottom graphs where the signal is sampled below the labium.

Another interesting observation regarding the signals sampled outside and inside a pipe can be made in the case of blowing speed 818 cm/s. This is a situation where the simulated 20 cm closed-end recorder fails to sing, probably because the outlet region above the labium is small and confined versus infinite in the physical world as explained in section 1.4. The signals sampled outside and inside the pipe for blowing speed 818 cm/s are shown in figures 7-8 and 7-9 respectively (figure 7-8 is the same

as figure 1-15 except for a longer interval of time). We can see that the density oscillations outside the pipe diminish quickly after 100×0.206 ms. However, there are periodic density oscillations inside the pipe at the frequency of 820 Hz.

The oscillations at frequency 820 Hz are most likely edge tones (hydrodynamic oscillations of the jet of air impinging the labium). It appears that the acoustic coupling between the jet and the pipe breaks down, and there is no strong amplification of sound. The frequency of an edge tone is proportional to the blowing speed approximately. Using the experimental equation 7.1 for edge tones, and putting $W = 0.4$ cm for the distance between the orifice and the labium, we find $f/V \sim 1\ cm$ which agrees with $f \sim 820$ Hz and $V = 818$ cm/s.

Another way of examining the oscillations at frequency 820 Hz is shown in figure 7-4 which plots iso-vorticity contours of the flue-labium region at 38.2 ms after startup (blowing speed 818 cm/s). We can see that the jet oscillates at blowing speed 818 cm/s even though little acoustic sound is produced by the recorder. However, the jet oscillations are relatively small compared to other situations when there is a strong acoustic signal. To compare, figure 7-5 shows the jet oscillations at blowing velocity 1104 cm/s and 34.7 ms after startup. Now, the jet oscillations are much larger than figure 7-4, and the vortices do not align themselves into a stream of vortices above the labium. The formation of a stream of vortices at blowing speed 818 cm/s is most-likely related to the small blowing speed and the absence of strong acoustic oscillations.

Figure 7-3 shows the jet oscillations of the 20 cm closed-end recorder at blowing speed 818 cm/s and 11.7 ms after startup. The acoustic signal is still strong at this time, the jet oscillations are large, and the shed vortices are not aligned into a stream of vortices. This happens later, approximately 20 ms after startup.

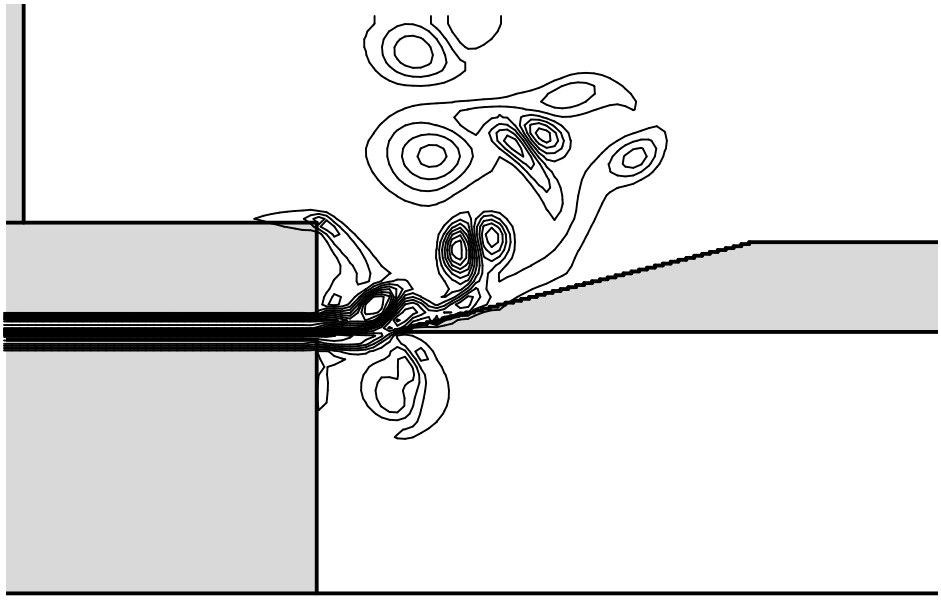

Figure 7-3: Simulation of 20 cm closed-end recorder, 11.7 ms after startup, blowing speed 818 cm/s, iso-vorticity contours.

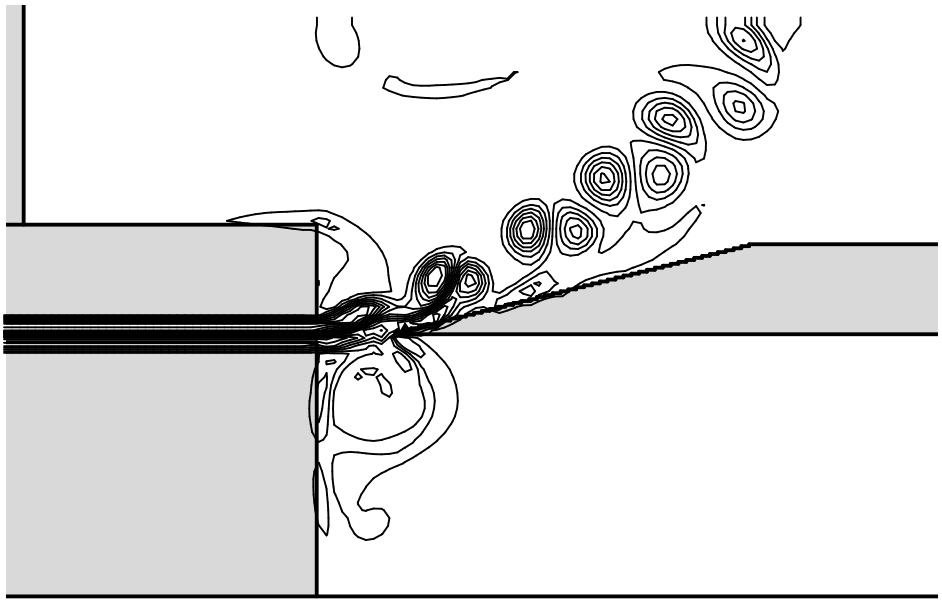

Figure 7-4: Simulation of 20 cm closed-end recorder, 38.2 ms after startup, blowing speed 818 cm/s, iso-vorticity contours. The jet oscillations are small and without acoustic amplification.

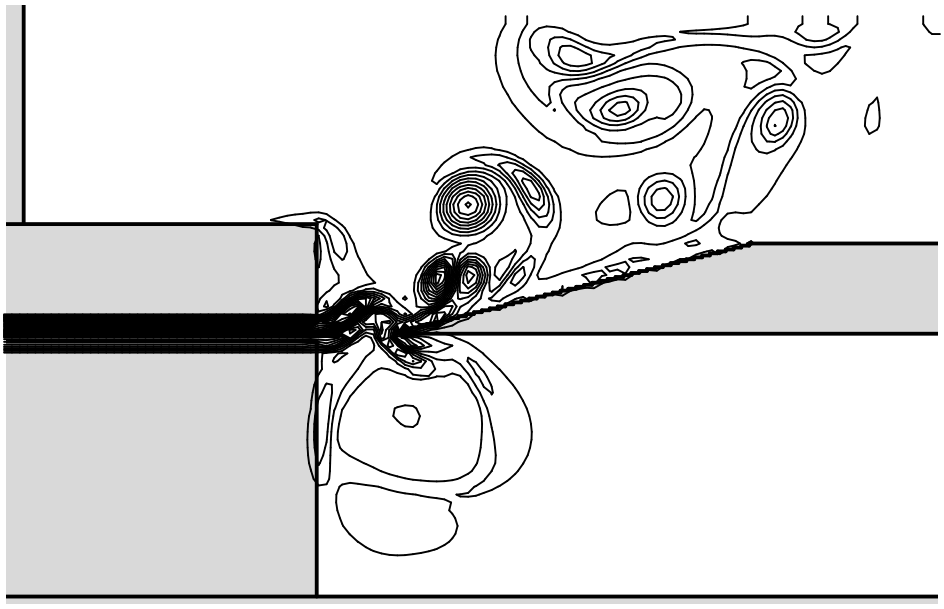

Figure 7-5: Simulation of 20 cm closed-end recorder, 34.7 ms after startup, blowing speed 1104 cm/s, iso-vorticity contours. The jet oscillations are large and produce strong acoustic waves.

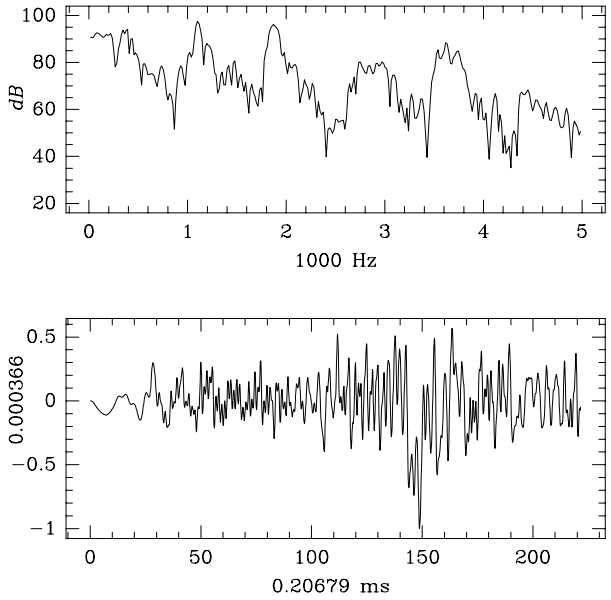

Figure 7-6: Lattice Boltzmann method, 20 cm closed-end soprano recorder, blowing velocity 1535 cm/s, sampled 5 cm above labium.

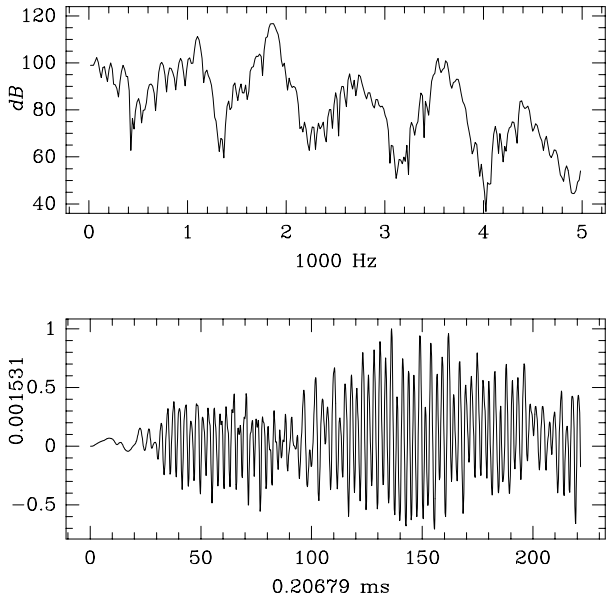

Figure 7-7: Lattice Boltzmann method, 20 cm closed-end soprano recorder, blowing velocity 1535 cm/s, sampled 1.34 cm below labium.

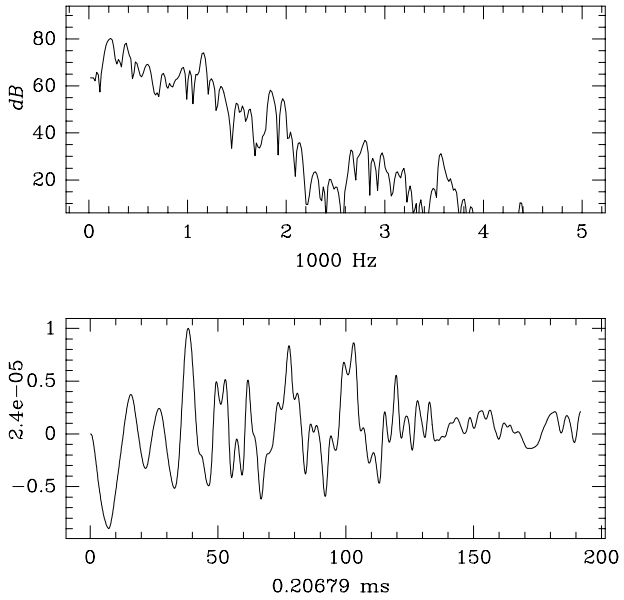

Figure 7-8: Lattice Boltzmann method, 20 cm closed-end soprano recorder, blowing velocity 818 cm/s, sampled 5 cm above labium.

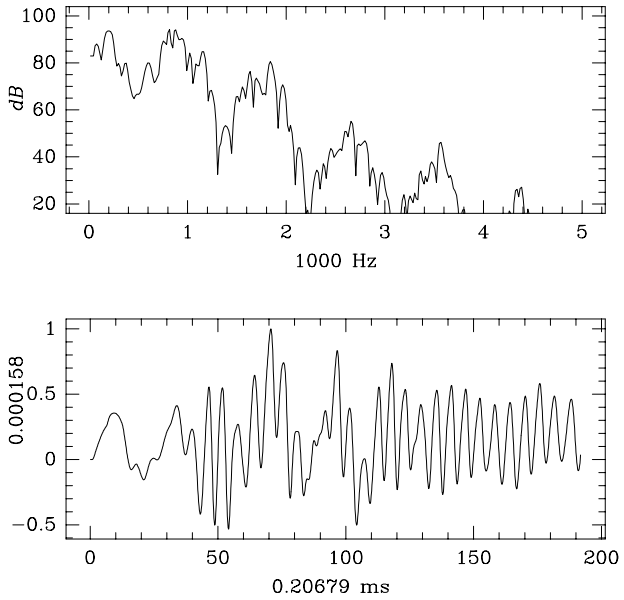

Figure 7-9: Lattice Boltzmann method, 20 cm closed-end soprano recorder, blowing velocity 818 cm/s, sampled 1.34 cm below labium. An edge tone perhaps occurs here.

7.5 Open-end soprano recorder

An open-end version of the soprano recorder is examined here. The geometry is the same as the one described in section 1.4 except for one difference. Here, the head of the recorder is connected to a pipe which is open at the distant end. Also, the total length of the pipe (including the head of the recorder) is chosen to be 22 cm in the present experiments. The frequencies generated by the open-end recorder are expected to be in ratios of $1 : 2 : 3 : 4$ in contrast to the ratios $1 : 3 : 5 : 7$ for a closed-end recorder. An open-end recorder behaves like an open-open pipe because there is one opening above the labium and another opening at the far end of the pipe. The computer simulations of the 22 cm open-end recorder confirm this behavior as we shall see below.

The boundary conditions at the open-end pipe are set according the scheme described in section 7.3; namely, the density is held constant (ambient pressure), and the velocity is extrapolated from the nearest neighboring node in the horizontal direction (normal to the open end). The boundary conditions at the inlet (flue channel) and the outlet (above the labium) are set in the same way as for a closed-end pipe; namely, both the density and the incoming/outgoing velocity are imposed.

A complication arises with the balance of incoming flow and outgoing flow because there is outgoing flow both through the top outlet and through the open-end pipe. In the present simulations using the lattice Boltzmann method (figures 7-13 and 7-14), the imposed outgoing flow at the top outlet has been set equal to the imposed incoming flow at the flue inlet. However, the imposed pressure drop has been set large enough that the actual incoming flow through the flue channel is significantly larger than the imposed flow (similar idea as in table 7.1 of section 7.3). This produces adequate incoming flow to balance both the flow through the top outlet and the flow through the open-end pipe. Experimentally, it has been measured that the time-average flow of air through the open-end pipe is about $1/10$ of the incoming flow through the flue channel, and that the remaining $9/10$ of the incoming flow exits

through the top outlet. In future simulations, it would be a good idea to set the imposed inflow at the inlet proportional to 10/10, and the imposed outflow at the outlet proportional to 9/10, so that 1/10 is left to flow through the open-end pipe. [4]

Figures 7-13 and 7-14 show the acoustic signal from simulations of the 22 cm open-end recorder sampled outside and inside the pipe. The major frequencies are summarized in table 7.2. For comparison, physical measurements of the acoustic signal of a 22 cm open-end recorder are shown in figures 7-15 and 7-16 and table 7.3. Table 7.4 lists the ideal frequencies of a passive pipe which is 22 cm long. The blowing velocity was not measured in the physical experiments, but it is estimated that the velocity was on the order of 1000-1500 cm/s (a human subject blew the recorder in these measurements).

Figure 7-10 plots iso-vorticity contours of the flue-labium region 38.2 ms after startup for the 22 cm open-end recorder at blowing speed 1197 cm/s. Comparing this figure against figure 7-5 of a closed-end recorder, we see that the oscillations of the jet extend inside the pipe in the case of an open-end recorder. Furthermore, large vortices are shed inside the pipe (below the labium) as well as outside the pipe. This behavior can also be seen in figures 7-11 and 7-12 which show a sequence of frames of the flue-labium region at 29.5 ms after startup. The frames are 0.2169 ms apart. The top figure shows the velocity vector field, and the bottom figure shows iso-vorticity contours. [5]

[4] In the present simulation of the 22 cm open-end recorder, the imposed influx is 1080 cm/s and imposed pressure drop is 6.48×10^6 gm/(cm s^2) divided by the mean density of air. The resulting incoming flow is 1197 cm/s, and the resulting pressure drop is approximately 2.55×10^6 in the same units as above. The resulting pressure drop is measured by examining the time-average density at points near the inlet and the top outlet (about 1 cm away from the boundaries), and by measuring the density gradient inside the flue channel to calculate the pressure drop along the full length of the flue channel.

[5] The contours of figure 7-12 are not as nice and smooth as the contours of, say, figure 1-11 because the present data was saved on disk at 4 times lower resolution than figure 1-11.

Vmean	f_0	(λ_0)	A_0	f_1	(λ_1)	A_1	f_2	(λ_2)	A_2	f_3	(λ_3)	A_3
cm/s	Hz	(cm)	10^{-5}	Hz	(cm)	10^{-6}	Hz	(cm)	10^{-6}	Hz	(cm)	10^{-6}
1197	1321	(26)	0.99	667	(52)	5.75	780	(44)	1.93	1512	(23)	1.84

Table 7.2: Frequencies, lattice Boltzmann, 22 cm open-end recorder

Vmean	f_0	(λ_0)	A_0	f_1	(λ_1)	A_1	f_2	(λ_2)	A_2	f_3	(λ_3)	A_3
cm/s	Hz	(cm)	10^{-1}	Hz	(cm)	10^{-2}	Hz	(cm)	10^{-3}	Hz	(cm)	10^{-3}
	691	(50)	6.39	1381	(25)	2.24	2071	(17)	0.97	2761	(12)	0.468

Table 7.3: Frequencies, physical measurements, 22 cm open-end recorder

22 cm pipe	f_0	(λ_0)	f_1	(λ_1)	f_2	(λ_2)	f_3	(λ_3)	f_4	(λ_4)
	Hz	(cm)	Hz	(cm)	Hz	(cm)	Hz	(cm)	Hz	(cm)
open-closed	391	(88)	1173	(29)	1955	(18)	2736	(13)	3518	(10)
open-open	782	(44)	1564	(22)	2345	(14.7)	3127	(11)	3909	(8.8)

Table 7.4: Ideal resonant frequencies, 22 cm, open-closed and open-open.

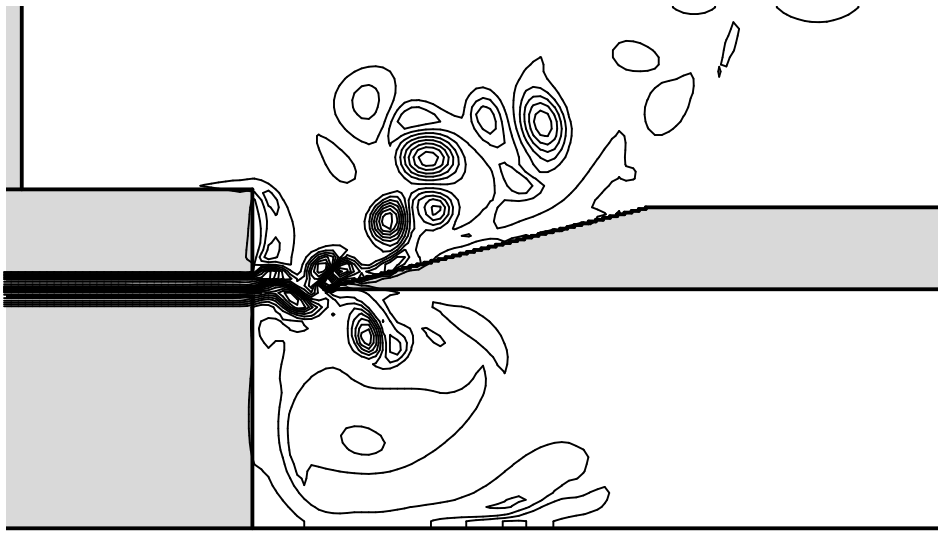

Figure 7-10: Lattice Boltzmann simulation of 22 cm open-end recorder, 31.2 ms after startup, blowing speed 1197 cm/s, iso-vorticity contours.

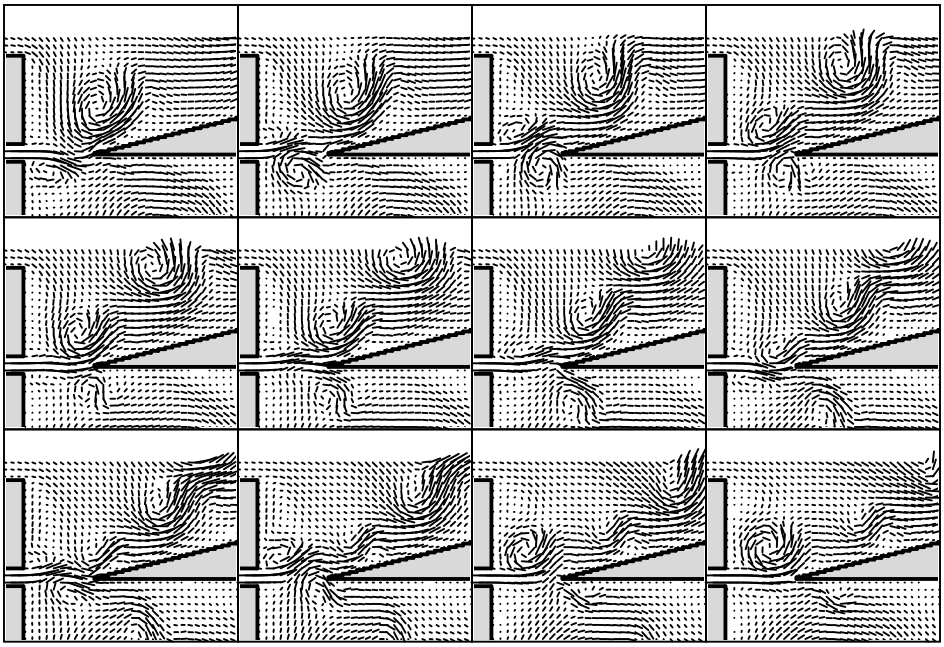

Figure 7-11: Frames left to right 0.2169 ms apart, velocity vector field, 22 cm open-end recorder, 29.5 ms after startup, blowing speed 1197 cm/s.

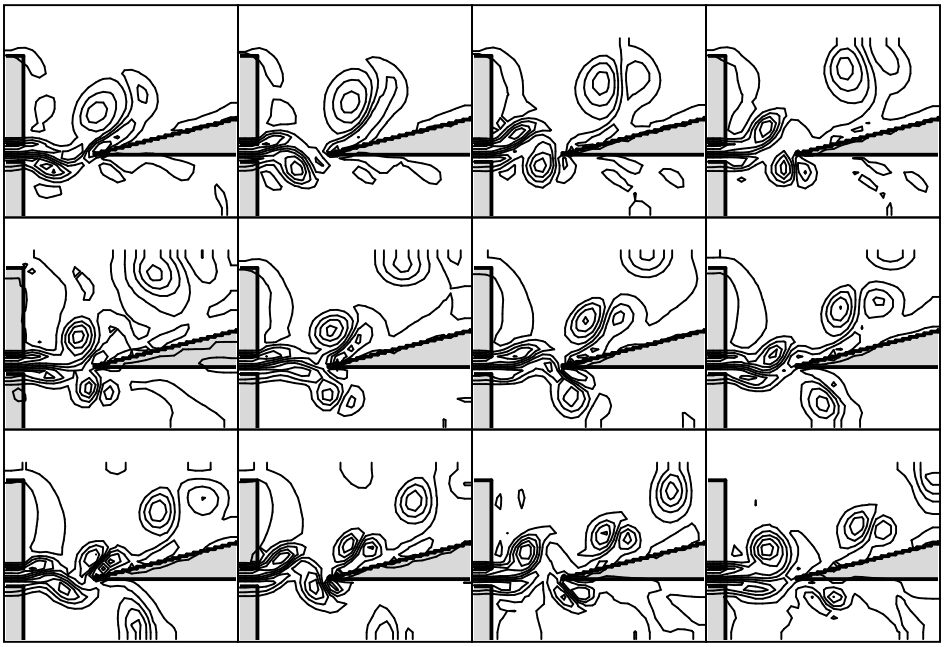

Figure 7-12: Frames left to right 0.2169 ms apart, iso-vorticity contours, 22 cm open-end recorder, 29.5 ms after startup, blowing speed 1197 cm/s.

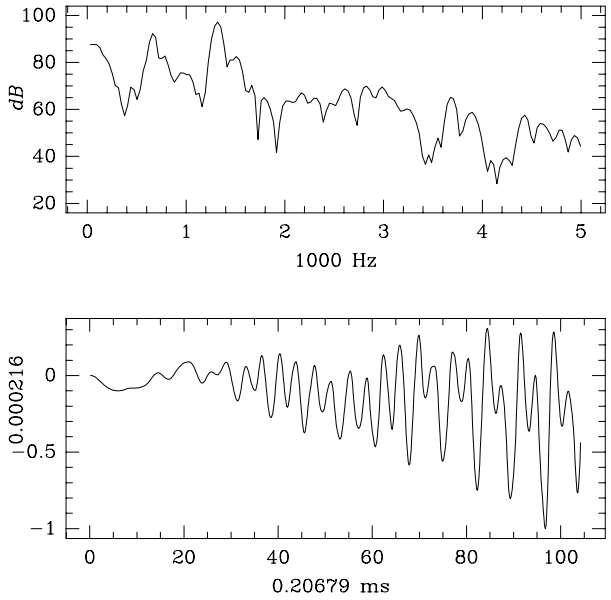

Figure 7-13: Lattice Boltzmann method, 22 cm open-end soprano recorder, blowing velocity 1197 cm/s, sampled 5 cm above the labium.

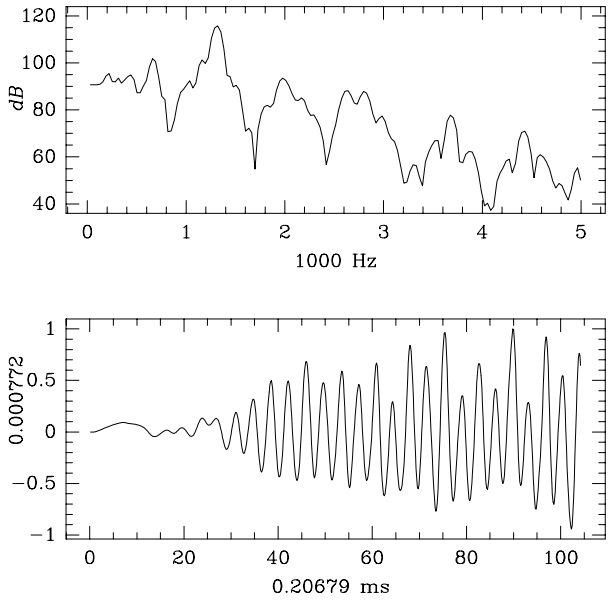

Figure 7-14: Lattice Boltzmann method, 22 cm open-end soprano recorder, blowing velocity 1197 cm/s, sampled 1.34 cm below the labium.

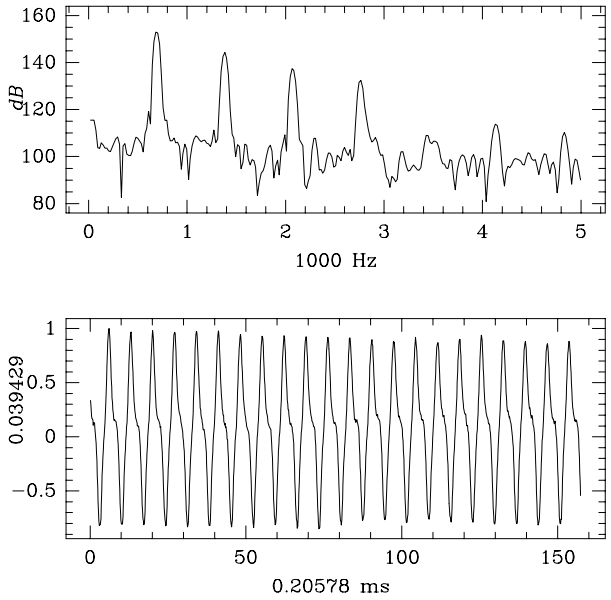

Figure 7-15: Physical measurements, 22 cm open-end recorder, steady state. Arbitrary units of amplitude.

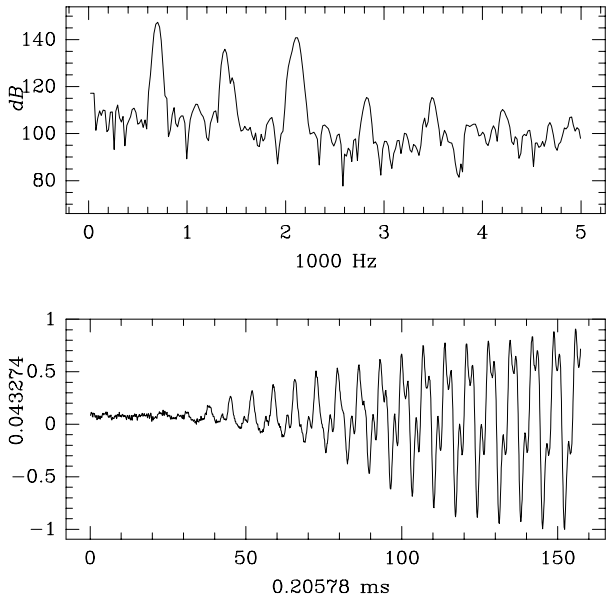

Figure 7-16: Physical measurements, 22 cm open-end recorder, startup transient.

Chapter 8

Conclusion

8.1 What has been accomplished

> After all the work is done, comes the point when we ask,
> Are we any better off than when we started ?

I think that the answer is "YES" in a number of ways. First, the big picture is that a previously unexplored area of fluid dynamics has succumbed to computer simulation. Using parallel computing on a cluster of non-dedicated workstations, the first simulations of hydrodynamics and acoustic waves inside wind musical instruments have been performed. Further, the simulations are in reasonable agreement with physical measurements of the acoustic signal of various flue pipes. Prior to my thesis, there were doubts whether the simulation of flue pipes using the compressible Navier Stokes equations is feasible. Some of the difficulties which seemed un-surmountable are the following.

- Whether enough compute cycles can be found (very small integration time steps must be used).
- Whether two-dozen non-dedicated workstations in my research group can be harnessed to perform intensive parallel computing for days and weeks without

disturbing the regular users.

- Whether the numerical stability problems (slow-growing high-frequency oscillations) which arise in simulations of subsonic compressible flow can be handled.
- Whether the lattice Boltzmann method (one of the numerical methods I use) can work at all.
- Whether uniform grids can be successful in simulating the sharp edge (labium) of a flue pipe.

My thesis has not found the best solutions to these problems, but has found some good-enough solutions, and this is the first step.

The approach presented here can be easily applied to other problems. In particular, the numerical techniques of my thesis are generally applicable to any flow problem of compressible subsonic flow. Also, the programming techniques and the organization of my parallel simulation system on a cluster of non-dedicated workstations can be applied to any problem that involves local-interactions and a static decomposition (vision problems, for example). My parallel system is very simple and effective because the constraints of local and static problems have been fully exploited.

One of the messages of my thesis is that a cross-disciplinary approach is needed for solving problems in scientific computing. The mathematics, the numerical modeling, the parallelization, the low-level system implementation, the sharing of the workstations, the different software abstractions and the representations of the problem, and many other issues have all been considered together more-or-less in order to find good effective solutions. In other words, my thesis promotes a generalist's approach.

Another message of my thesis is that explicit methods are very promising for parallel computing. In the present simulations, there is a match between the requirements of the problem (small time steps for subsonic compressible flow), the requirements of explicit methods, and the requirements of the computer system (small communication capacity on a cluster of workstations). In general, however, explicit methods are

desirable for parallel computing when increasing numbers of local processing units are available with small communication capacity between the processing units. Perhaps, future parallel computers will consist of millions of local processing units, each unit having the power of one of today's workstations. Communication is going to dominate the cost of such computers, and methods that minimize communication are going to be desirable. A vision of such immense computers has guided many of the approaches of my thesis.

Apart from the big picture, my thesis has also numerous detailed results to offer. One result is the demonstration and the analysis of artificial-viscosity filters for mitigating the high-frequency instabilities of subsonic compressible flow. Another numerical result is my work on the boundary conditions and the accuracy of the lattice boltzmann method. With regard to distributed computing, the simple structure of my program, and the automatic process migration are worth remembering. With regard to the physics of musical instruments, the detailed pictures of the jet of air oscillating inside a flue pipe are unique and very important for studying this complex phenomenon.

Directions for future work are summarized below.

8.2 Ideas for future work

8.2.1 Physical Applications

- Someday soon, it may be possible for the computer to find reduced models of flue pipes automatically (see section 7.2 for an introduction to reduced models of flue pipes) by performing a few preliminary direct simulations of flue pipes, and then examining the results. The present simulation system could be combined with another "intelligent" program which knows about a number of possible reduced models, and tries to fit the best model to the data.

- The present simulations can be easily extended to include flue pipes with finger holes, and also pipes which are simple models for the human vocal tract. (see Shadle [46] for a simple pipe that models the vocal tract).

- The present approach of simulating compressible subsonic fluid dynamics has applications in the design of oil and gas carrying pipes [37], and perhaps in the study of medical issues such as the acoustic waves inside blood arteries and non-intrusive measurements of arteriosclerosis [24], etc.

- New applications may arise in the future. For example, undulating jets of burning fuel may be able to increase the efficiency of a combustion engine. This is a very distant idea at present, but it deserves some attention. Computer simulations such as the ones presented here will be very important in such future studies. The present simulations must be extended to model heat conduction and two-phase flow.

8.2.2 Parallel computing

- We have seen that explicit methods are highly-suitable for parallel computing, but require very small time steps for stability. Between implicit methods (full matrix equation) and explicit methods (local-interactions) there may exist intermediate methods; for example, methods that use small matrices that do not extend the full length of the numerical grid. Such methods might lead to improved numerical stability while preserving the benefits of local-interaction algorithms (see section 3.2).

- Uniform grids such as the ones employed here are very simple and work well, but they are not very efficient. Non-uniform grids are needed in order to focus the computational power on regions where it is mostly needed such as sharp obstacles. Unstructured non-uniform grids are very promising, and a lot of

research is currently being done on them [6]. An interesting project is to try to develop unstructured grids on a cluster of non-dedicated workstations.

8.2.3 Numerical analysis

- Section 5.5 raises some interesting questions regarding the relationship between artificial-viscosity filters, physical turbulence, and perhaps a kind of "discrete turbulence" which is a property of systems of difference equations as opposed to differential equations.

- There is a need to develop numerical conditions that approximate an infinite region at the outlet boundary (see section 7.3), and also suitable techniques that remove the generated vorticity from the simulated region in order to continue the simulations of flue pipes for indefinitely long periods of time (see sections 1.4 and 7.3).

- A comprehensive theoretical analysis of the stability and the accuracy of the lattice Boltzmann method is incomplete at the present time (see section 4.1.3).

List of Figures

List of Tables

Bibliography

[1] Henri Bal, Frans Kaashoek, and Andrew Tanenbaum. Orca: A languate for parallel programming of distributed systems. *IEEE Transactions on Software Engineering*, 18(3):190–205, March 1992.

[2] Z. Baolin and L. Wenzhi. On alternating segment crank-nicolson scheme. *Parallel Computing*, 20:897–902, 1994.

[3] G.K. Batchelor. *An Introduction to Fluid Dynamics.* Cambridge University Press, 1967.

[4] P. Bhatnagar, E.P. Gross, and M.K. Krook. A model for collision processes in gases. *Physical Review*, 94:511, 1954.

[5] R.D. Blumofe and D.S. Park. Scheduling large-scale parallel computations on networks of workstations. In *Proceedings Of High Performance Distributed Computing 94, San Franscisco, California*, pages 96–105, 1994.

[6] W.J. Camp, S.J. Plimpton, B.A. Hendrickson, and R.W. Leland. Massively parallel methods for engineering and science problems. *Communications of the ACM*, 37(4):31–41, April 1994.

[7] C. H. Cap and V. Strumpen. Efficient parallel computing in distributed workstation environments. *Parallel Computing*, 19(11):1221–1234, 1993.

[8] N. Carriero, D. Gelernter, D. Kaminsky, and J. Westbrook. *Adaptive Parallelism with Piranha.* Report No. YALEU/DCS/RR-954, Department of Computer Science, Yale University, February 1993.

[9] J.S. Chase, F.G. Amador, E.D. Lazowska, H.M. Levy, and R.J. Littlefield. The Amber system: Parallel programming on a network of multiprocessors. *ACM SIGOPS Operating Systems Review*, 23(5):147–158, 1989.

[10] H. Chen, S. Chen, and W.H. Matthaeus. Recovery of the Navier-Stokes equations using a lattice-gas Boltzmann method. *Phys. Rev. A*, 45(8):5339–5342, April 1992.

[11] G. Chesshire and V.K. Naik. An environment for parallel and distributed computation with application to overlapping grids. *IBM Journal Research and Development*, 38(3):285–300, May 1994.

[12] R. Cornubert, D. d'Humières, and D. Levermore. A knudsen layer theory for lattice gases. *Lattice Gas Methods For Pde's: Theory, Application, And Hardware, Physica D,*, 47:241–259, 1991.

[13] R. Courant. *Differential and Integral Calculus, Volume II.* Wiley Interscience, 1938.

[14] R. Courant, K.O. Friedrichs, and H. Lewy. On the partial difference equations of mathematical physics. *IBM Journal of Research and Development*, 11:215–234, 1967.

[15] D. d'Humières and P. Lallemand. Numerical simulations of hydrodynamics with lattice gas automata in two dimensions. *Complex Systems*, 1:599–632, 1987.

[16] Fredirick Douglis. *Transparent Process Migration in the Sprite Operating System.* Report No. UCB/CSD 90/598, Computer Science Division (EECS), University of California Berkeley, September 1990.

[17] Molvig et al. *Unpublished reports.* MIT Department of Nuclear Engineering, 1991.

[18] C.A.J. Fletcher. *Computational Techniques For Fluid Dynamics*, volume 1. Springer-Verlag, New York, N.Y., 1988.

[19] G. Fox, M. Johnson, G. Lyzenga, S. Otto, J. Salmon, and D. Walker. *Solving Problems on Concurrent Processors*, volume 1. Prentice-Hall Inc., 1988.

[20] U. Frisch, D. d'Humières, B. Hasslacher, P. Lallemand, Y. Pomeau, and J.-P. Rivet. Lattice gas hydrodynamics in two and three dimensions. *Complex Systems*, 1:649–707, 1987.

[21] I. Ginzbourg and P. Adler. Boundary flow condition analysis for the three-dimensional lattice boltzmann model. *Journal de Physique II*, 4:191–214, 1994.

[22] A. K. Gunstensen, D. H. Rothman, S. Zaleski, and G. Zanetti. Lattice Boltzman model of immiscible fluids. *Phys. Rev. A*, 43(8):4320–4327, April 1991.

[23] A.K. Gunstensen and D.H. Rothman. A galilean-invariant immiscible lattice gas. *Physica D*, 47:53–63, 1991.

[24] J.C. Harding and D.S. Pope. Sound generation by a stenosis in a pipe. *AIAA Journal*, 30(2):312–317, February 1992.

[25] F.J. Higuera and J. Jimenez. Boltzman approach to lattice gas simulations. *Europhys. Lett.*, 9(7):663–668, 1989.

[26] A. Hirschberg. *Wind Instruments.* Eindhoven Institute of Technology, Report R-1290-D, 1994.

[27] K. Huang. *Statistical Mechanics.* Second Edition, John Wiley and Sons, New York N.Y., 1987.

[28] C. Kittel and H. Kroemer. *Thermal Physics.* Second Edition, W.H. Freeman and Company, New York, 1980.

[29] J.M.V.A. Koelman. A simple lattice Boltzman scheme for Navier Stokes fluid flow. *Europhys. Lett.*, 15(6):603–607, 1991.

[30] S. Kohn and S. Baden. *A robust parallel programming model for dynamic non-uniform scientific computations.* Report CS94-354, University of California, San Diego, 1994.

[31] H. Lamb. *Hydrodynamics.* Sixth Edition, Dover Publications, N.Y., 1932,1945.

[32] L.D. Landau and E.M. Lifshitz. *Fluid Mechanics, 2nd Edition.* Pergamon Press, New York, NY, 1989.

[33] P.M. Morse and U.K. Ingard. *Theoretical Acoustics.* McGraw-Hill, New York, NY, 1968.

[34] J.N. Newman. *Marine Hydrodynamics.* The MIT Press, Cambridge, MA, 1977,1986.

[35] S. Ohring. Calculations of self-excited impringing jet flow. *Journal of Fluid Mechanics*, 163:69–98, 1986.

[36] H.F. Olson. *Music, Physics, and Engineering.* Second Edition, Dover Publications Inc, New York, 1967.

[37] M.C.A.M. Peters. *Aeroacoustic Sources in Internal Flows.* Eindhoven University of Technology, Ph.D. Dissertation, 1993.

[38] R. Peyret and T. D. Taylor. *Computational Methods For Fluid Flow.* Springer-Verlag, New York, N.Y., 1990.

[39] T.J. Poinsot and S.K. Lele. Boundary conditions for direct simulations of compressible viscous flows. *Journal of Computational Physics*, 101:104–129, 1992.

[40] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. *Numerical Recipes In C.* Cambridge University Press, New York, N.Y., 1988.

[41] Y.H. Qian, D. d'Humières, and P. Lallemand. Lattice BGK models for Navier Stokes equation. *Europhys. Lett.*, 17(6):479–484, 1992.

[42] J.W.S. Rayleigh. *The Theory of Sound.* volume 2, Second Edition, Dover Publications Inc, New York, 1945.

[43] R.D. Richtmeyer and K.W. Morton. *Stabilities studies with difference equations.* Courant Institute, New York University Rept. NY)-1480-5, 1964.

[44] H. Schlichting. *Boundary Layers Theory.* McGraw-Hill, New York, NY, 1968.

[45] R. Sekuler and R. Blake. *Perception.* Alfred A. Knopf, Inc., New York, N.Y., 1985.

[46] Christine H. Shadle. *The Acoustics of Fricative Consonants.* Department of Electrical Engineering and Computer Science MIT, Ph.D. Dissertation, 1985.

[47] W.M. Siebert. *Circuits, Signals, and Systems.* MIT Press and McGraw-Hill, 1986.

[48] P.A. Skordos. Initial and boundary conditions for the lattice Boltzmann method. *Physical Review E*, 48(6):4823–4842, December 1993.

[49] P.A. Skordos and W.H. Zurek. Maxwell's demon, rectifiers, and the second law: Computer simulation of smoluchowski's trapdoor. *American Journal of Physics*, 60(10):876–882, October 1992.

[50] V. S. Sunderam. A framework for parallel distributed computing. *Concurrency: Practice and Experience*, 2(4):315–339, December 1990.

[51] G. I. Taylor. On the decay of vortices in a viscous fluid. *Philosophical Magazine*, Series 6, 46:671–674, 1923.

[52] J.F. Thompson, J.L. Steger, and H. Yoshihara. *Three Dimensional Grid Generation For Complex Configurations – Recent Progress.* Agard-Ag-309, North Atlantic Treaty Organization, March 1988.

[53] T. Toffoli and N. Margolus. Programmable matter: concepts and realization. *Phsica D*, 47:263–272, 1991.

[54] D.J. Tritton. *Physical Fluid Dynamics.* Oxford Science Publications, Second Edition, 1988.

[55] M. Vergassola, R. Benzi, and S. Succi. On the hydrodynamic behavior of the lattice Boltzman equation. *Europhys. Lett.*, 13(5):411–416, 1990.

[56] M.P. Verge, R. Caussè, B. Fabre, A. Hirschberg, and A. van Steenbergen. Jet oscillations and jet drive in recorder-like instruments. *accepted for publication in Acta Acustica*, 1994.

[57] M.P. Verge, B. Fabre, W.E.A. Mahu, A. Hirschberg, R.R van Hassel, and A.P.J. Wijnands. Jet formation and jet velocity fluctuations in a flue organ pipe. *Journal of Acoustical Society of America*, 95(2):1119–1132, February 1994.

[58] S. Wolfram. Cellular automata fluids 1: Basic theory. *J. Stat. Phys.*, 45(3/4):471–526, 1986. Also Available In "Lattice Gas Methods For Partial Differential Equations", Ed. G.D. Doolen, Pp 19–73, (Sfi Sisoc Addison-Wesley 1990).